Universidade de Aveiro
2011
Departamento de Eletrónica, Telecomunicações e Informática

Diogo Gomes Pinho Silva Loureiro

Suporte de monitorização baseado em NETCONF

Universidade de Aveiro Departamento de Eletrónica, Telecomunicações e Informática
2011

Diogo Gomes Pinho Silva Loureiro

Suporte de monitorização baseada em NETCONF

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Professor Doutor António Manuel Duarte Nogueira, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Professor Doutor Pedro Alexandre de Sousa Gonçalves, Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro.

Dedico este trabalho aos meus pais e irmã pela inesgotável paciência e compreensão.

o júri

presidente

Prof. Dr. Rui Luís Andrade Aguiar
Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais

Prof. Dr. António Manuel Duarte Nogueira
Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (Orientador)

Prof. Dr. Pedro Alexandre de Sousa Gonçalves
Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda (Coorientador)

Prof. Dr. Carlos Manuel da Silva Rabadão
Professor adjunto do Departamento de Engenharia Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria

agradecimentos

Agradeço à Kinga, o apoio e paciência durante a realização deste trabalho, que nunca se negou a nenhum sacrifício em prol da sua conclusão.
Também aos meus coordenadores o meu sincero obrigado pela sua disponibilidade e apoio incondicional.

palavras-chave

Gestão de redes, Gestão de Configurações, NETCONF, YANG, Monitorização de redes.

Resumo

A necessidade de gestão dos equipamentos das redes tem juntado em volta de organismos normalizadores como o IETF e o DMTF, a comunidade académica e os fabricantes de equipamentos. A evolução das características das redes, como por exemplo a sua dimensão, o número e a heterogeneidade dos equipamentos interligados, e a crescente diversidade de serviços de rede têm vindo a alterar os requisitos de gestão e, por conseguinte, a criar a necessidade de novas tecnologias para gerir essas redes. A tecnologia de gestão SNMP surgiu em meados dos anos 80 e, apesar de um conjunto de defeitos que rapidamente lhe foram apontados, rapidamente se tornou a tecnologia de gestão de facto, sendo omnipresente na maioria dos equipamentos de rede e estando disponível sob a forma de imensas APIs e aplicações de gestão. Sendo uma tecnologia nascida de entre a comunidade de gestão de redes IP, não incluía outros detalhes relacionados com a gestão de sistemas e serviços que entretanto foram incluídos pelo DMTF na tecnologia WBEM, segundo uma lógica de gestão integrada. O WBEM inclui já tecnologias da web para representar e codificar a informação de gestão, de forma a fomentar a interoperabilidade da gestão dos equipamentos de diferentes modelos e fabricantes. Com o advento dos Web services, e dada as suas vantagens de rápido desenvolvimento e interoperabilidade, as entidades normalizadoras da área da gestão dos sistemas propuseram novas tecnologias como o WSDM-MUWS do OASIS e o WS-MAN do DMTF. Como forma de ultrapassar os problemas desde sempre apontados ao SNMP, especialmente os relacionados com a sua segurança e falta de escalabilidade para transporte de grandes quantidade de informação, o IETF desenvolveu uma nova tecnologia designada de NETCONF que utiliza a codificação XML e alternativas de transporte de informação seguras e fiáveis. Normalizou também uma linguagem para descrição da informação de gestão, o YANG, criada especificamente para ser utilizada com este protocolo. Neste trabalho, implementou-se uma solução de monitorização utilizando a tecnologia NETCONF, que efetua o transporte da informação de gestão em SOAP. A presente dissertação documenta a implementação da solução de monitorização NETCONF proposta e da respetiva avaliação, comparado as características e capacidades da tecnologia utilizada com as tecnologias de gestão SNMP e WBEM em termos de tráfego gerado, de eficiência de sinalização e de tempos de resposta. Da análise destes testes são tiradas ilações acerca do desempenho destes protocolos e da viabilidade do NETCONF como solução futura para a gestão e monitorização de redes.

keywords

Network Management, Configuration Management, NETCONF, YANG, Network Monitorization.

Abstract

The need for management of network equipment has gathered around standard setting bodies like the IETF and the DMTF, the academic community and equipment manufacturers. The evolution of network characteristics such as its size, the number and heterogeneity of devices, and the growing diversity of network services are changing the management requirements and, therefore creating the need for new technologies to manage these networks. The SNMP management technology emerged in the mid 80s and, despite a number of defects that were pointed out, it quickly became the *de facto* management technology, is ubiquitous in most network equipment and is available in lots of APIs and management applications. Being a technology born from IP network management community, it did not include other details related to the management of systems and services which have been included in the DMTFs WBEM standard, for integrated management. WBEM already includes web-based technologies to represent and encode management information in order to enhance interoperability among the solutions and equipment from different manufacturers. With the advent of Web services, and given its advantages of rapid development and interoperability, entities standardizing management systems proposed new technologies such as the OASIS WSDM-MUWS and the DMTF WS-MAN. To overcome the problems pointed to SNMP, especially those related to its safety and lack of scalability to transport large amount of information, the IETF has developed a new technology called NETCONF that uses the XML encoding and several alternatives for secure and reliable transport of information. They also normalized a language for describing management information, YANG, created specifically for use with this protocol. In this work, we implemented a monitoring solution using NETCONF, which makes the transport of management information in SOAP. This dissertation documents this implementation, the relevant technical assessment of the proposal and compared the features and capabilities of the technology used with the WBEM and SNMP technologies in terms of generated traffic, coding efficiency and response times. From the analysis of these tests lessons are taken about the performance of these protocols and the feasibility of NETCONF as a solution for the future of network management and monitoring.

Índice

Índice de Figuras

Índice de Tabelas

Caixas de Código

Lista de Acrónimos

ASN.1	Abstract Syntax Notation One
BCD	Boot Configuration Data
BEEP	Blocks Extensible Exchange Protocol
CIM	Common Information Model
CIMON	CIM Object Manager
COPS	Commom Open Policy Service
CQL	CIM Query Language
DMTF	Distributed Management Task Force
DSDL	Document Schema Defenition Language
DTD	Document Type Defininition
EPR	Endpoint References
FCAPS	Fault Configuration Accountting Performance Security
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IAB	Internet Architecture Board
IETF	Internet Engineering Task Force
IRTF	Internet Research Task Force
ISO	International Organization for Standardization
ITU-T	International Telecommunications Union, Standardization Sector
MIB	Manage Information Base
MIB	Management Information Base
MOF	Management Object Format
MTU	Maximum Transfer Unit
NETCONF	Network Configuration
NMS	Network Management System
OID	Object Identifier
OMG	Object Management Group
OSI	Open System Interconnect
OSPF	Open Shortest Path First
PDA	Personal Digital Assistant

PIB	Policy Informatio Base
RAM	Random Access Memory
RFC	Request For Comments
RPC	Remote Procedure Call
SDK	Software Developer Kit
SGR	Sistemas de Gestão de Redes
SMI	Structure of Management Information
SMIng	SMI next generation
SNMP	Simple Network Management Protocol
SO	Sistema Operativo
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Socket Layer
TLS	Transport Layer Security
UML	Unified Modeling Language
URI	Universal Resource Identifier
W3C	World Wide Web Consortium
Web UI	Web User Interface
WEBM	Web-Based Management
WMI	Windows Management Instrumentation
WQL	WBEM Query Language
WS	Web Service
WSDL	Web Service Description Language
WS-MAN	Web Services for Management
XML	Extensible Markup Language
XPath	XML path language
XSLT	Extensible Stylesheet Language Transformations

1. Introdução

Desde há muito que as redes de comunicação fazem parte do nosso quotidiano e servem de suporte às necessidades mais básicas das diferentes áreas da sociedade, como por exemplo infraestruturas governamentais, sistemas integrados de proteção civil, indústria e educação. Inerente ao crescente desenvolvimento das redes de comunicação, está o seu aumento em tamanho e complexidade. Estas redes ligam centenas e às vezes milhares de elementos de hardware e software, muitas vezes dispersos por grandes áreas geográficas. Como tal, é inevitável que surjam falhas na operação de equipamentos, que existam elementos mal configurados e a necessitar de ser revistos, recursos mal distribuídos na rede, entre outros. Assim, um administrador cuja função é manter a rede a funcionar otimizando os seus recursos, necessita de ferramentas que a permitam monitorizar, gerir e controlar [1].

A fim de definir melhor o que é a gestão de redes, o *International Organization for standardization* (ISO), em conjunto com o departamento de normas do *International Telecomunications Union* (ITU-T), definiu um modelo da gestão de redes, conhecido por FCAPS [2] (**Figura 1**), que distingue cinco áreas chave para Sistemas de Gestão de Redes (SGR):

Performanace Management – Lida com a quantificação, descrição, análise e controlo do desempenho (por exemplo, a taxa de transferência) dos diferentes elementos de uma rede,
Fault Management – Deve detetar, registar e notificar sobre eventuais falhas que ocorram nos equipamentos ou prestação de serviços da rede,
Configuration Management – Permite ao administrador da rede manter conhecimento sobre quais os elementos que fazem parte da rede gerida e quais as configurações existentes nesses dispositivos,
Accounting Management – Torna possível especificar e controlar os utilizadores, definir permissões de acesso e manter registo dos mesmos,
Security Management – Controla o acesso aos recursos da rede, tendo em conta as regras definidas no *Accounting Management*, para cada utilizador.

Ao longo dos anos tem havido um esforço por parte de várias organizações, como o *Open Systems Interconnect* (OSI) - *System Management*[3], ITU-T [4], *Internet Engineering Task Force* (IETF) [5], *Object Management Group* (OMG) [6] ou o *Internet Research Task Force* (IRTF) [7] para criar normas e protocolos que servissem o propósito da gestão de redes. Como será explicado mais à frente, existiram algumas ferramentas, como o *Simple Network Management Protocol* (SNMP) [8], que durante muitos anos foram satisfazendo as necessidades dos administradores de redes mas, com o crescimento

em tamanho e complexidade das redes e crescente aumento do número de serviços existentes, tornou-se necessário criar novas normas para dar resposta aos novos requisitos. Esses requisitos são normalmente de natureza funcional, mas também podem ser de índole normativa ou mesmo relacionada com o mercado [9].

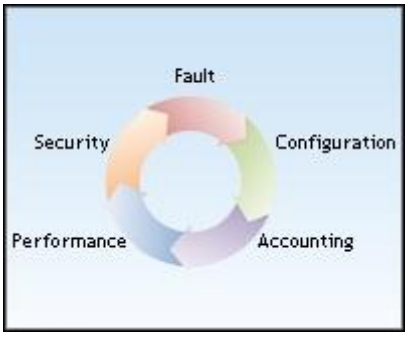

Figura 1. Esquema ilustrativo do modelo FCAPS para gestão de redes

Em 2002 o *Internet Architecture Board* (IAB) [10] promoveu um workshop [11], onde juntou operadores de telecomunicações, pessoas responsáveis por desenvolver as normas e protocolos e programadores de soluções finais de gestão de rede, com a intenção de desenvolver diretrizes para a definição futura de uma nova norma de gestão de redes pelo IETF. O workshop teve dois pontos principais: o primeiro era identificar as tecnologias existentes para gestão de redes e debater as respetivas vantagens e desvantagens, o segundo era identificar as reais necessidades dos gestores de redes.

Foram identificadas o SNMP, o *Common Open Policy Service* (COPS)[12] e o *Common Information Model* (CIM) [13] como as principais tecnologias existentes para gestão de redes. O SNMP, criado nos anos 80, embora muito utilizado começava a não acompanhar a evolução das redes de comunicação [11], revelando alguns problemas ao lidar com configurações cada vez maiores [14] e uma grande complexidade na implementação das *Model Information Base* (MIB) [15]. O protocolo COPS, definido no fim dos anos 90, apresentava uma grande complexidade na visualização de configurações e sempre foi visto como uma melhoria do SNMP, o que levou a uma falta de confiança por parte da industria visivél também pela pouca quantidade de *Policy Information Bases* (PIB) normalisadas. Por fim, o CIM foi desenvolvido pelo *Distributed Management Task Force* (DMTF) [16] em meados dos anos 90 com problemas de interoperabilidade entre *schemas* e entre as diferentes implementações. A **Figura 2** apresenta um cronograma onde é visivél o intervalo temporal em que o SNMP foi a principal alternativa para gestão e monitorização de redes IP, tendo posteriormente surgido alternativas em intervalos de tempo bem menores o que evidência a preocupação em encontrar soluções de gestão e monitorização que substituissem o SNMP e que acompanhassem a evolução das rede de comunicação.

Como linhas orientadoras dos requisitos funcionais identificados pelos operadores de telecomunicações para a gestão de redes, concluiu-se que:

- Era essencial permitir a distinção entre dados de configuração, tipicamente estáticos, e dados de estado, de natureza mais dinâmica,
- Um Sistema de Gestão de Redes (SGR) deveria ser simples de usar por parte dos operadores que deveriam poder configurar toda a rede em vez de configurar um só equipamento,
- Deveriam ser suportados mecanismos que permitissem operar configurações completas e era desejável o suporte para operar partes das configurações, em oposição ao suporte exclusivo dos objetos individuais das configurações,
- O protocolo deveria suportar a gestão de várias configurações num só dispositivo e a respetiva ativação. Deveria também ser possível fazer o *rollback* das configurações,
- Deveria ser possível ativar configurações de forma coordenada em vários dispositivos, para evitar inconsistência das configurações e simplificar significativamente as tarefas de gestão,
- As configurações deveriam existir num formato simples de ler por humanos, deveriam poder ser editadas por processadores de texto comuns e geridas por sistemas de controlo de versões,
- Os protocolos de gestão de redes deveriam permitir transporte seguro das comunicações, autenticação e controlo de acesso integrável com uma infraestrutura existente de gestão de chaves ou credenciais.

Existiam também várias questões relacionadas com o mercado que motivaram a avaliação de propostas para novas tecnologias de gestão de rede [17]. Um aspeto importante é o controlo que algumas empresas exercem sobre o negócio da gestão de redes. Enquanto algumas empresas contribuem para a criação de normas, outras mantêm soluções proprietárias de forma a fidelizar os clientes, o que dificulta a administração de redes com equipamentos de diferentes vendedores. Inicialmente um dos objetivos do SNMP era criar uma norma aberta para gestão de redes que criasse mais competitividade no mercado e levasse ao desenvolvimento de melhores aplicações. Mas o que se veio a verificar é que este modelo não funcionou como esperado, e o mercado de gestão de redes continua a ser controlado por soluções proprietárias onde existe um número muito reduzido de soluções para gerir redes de grande dimensão e complexidade, dominando as soluções de gestão individual de equipamentos. Outro aspeto relevante tem a ver com a decisão de compra, em que as funcionalidades de gestão de um equipamento pouco ou nada influenciam a decisão, dando-se prioridade às características tecnológicas e, claro está, ao preço. Devido à dificuldade por parte das empresas em calcular o custo de propriedade de um equipamento, normalmente opta-se por soluções inicialmente mais baratas, mesmo que mais tarde venha a ser necessário investir num novo sistema de gestão devido às incompatibilidades entre o sistema existente e os diferentes dispositivos. Consequentemente, como as soluções de gestão de redes não são um dos principais atrativos nos produtos, os vendedores descuraram no investimento em equipas para o

desenvolvimento dessas soluções, optando por investir em áreas com melhores resultados de marketing.

A juntar às questões funcionais e às questões de mercado, vêm as questões normativas. Uma das críticas mais comuns é referente aos longos ciclos para definição das normas, que podem por em causa a atualização das opções tecnológicas tomadas para uma determinada norma. Os longos períodos de negociação entre vendedores fazem também com que surjam antecipadamente soluções com problemas de compatibilidade, baseadas na que está em desenvolvimento. Outro problema diz respeito aos modelos de definição de dados. Os modelos de dados necessitam de ser atualizados à medida que se atualizam as tecnologias que eles suportam. No entanto, embora o SMI defina regras para atualizar as suas definições mantendo a interoperabilidade entre versões, as atualizações das MIBs surgem tardiamente ou não surgem de todo. Isto tanto por descuido do IETF na atualização dos modelos de dados das MIBs, como dos vendedores em atualizar as definições dos produtos, por causa do já mencionado, baixo retorno financeiro gerado por estas atualizações.

A forma como o SMI estrutura os dados, em tabelas, nunca gerou consenso. O cruzamento de dados de forma relacional e os tipos de dados limitados a valores escalares não satisfazia as necessidades dos administradores de redes. Uma abordagem com um modelo hierárquico é mais adequada para, por exemplo modelar os dados de uma interface ou tabelas de encaminhamento [18]. Embora com algum custo a nível de memória a representação hierárquica é mais legível e apresenta melhor desempenho em operações de filtragem [19]. E assim, surge o XML como opção de linguagem para modelação dos dados de gestão, exatamente por causa da sua representação hierárquica dos dados. No entanto, surgiu também nos anos 90 o *Distributed Mangement Task Force* (DMTF) que desenvolveu a linguagem *Common Information Model* (CIM) para descrição de sistemas de gestão, equivalente ao SMI, mas baseada num paradigma orientado a objetos. O DMTF foi também responsável por desenvolver uma norma de gestão, baseada em tecnologias Web, de redes equivalente ao SNMP, o *Web Based Enterprise Management* (WBEM).

Como consequência da necessidade de novas ferramentas para a gestão de redes surgiram, no workshop do IAB em 2002, diretrizes para criação de um novo protocolo por parte do IETF que tivesse em atenção as recomendações feitas e que utilizasse uma linguagem baseada em XML para descrição dos dados de gestão, o NETCONF. Posteriormente, em 2003 é criado o grupo de trabalho do NETCONF da secção de "*Operations and Managment*" do IETF[20]. Como resultado, em Dezembro de 2006 são aprovadas as primeiras quatro RFCs referentes ao NETCONF, a RFC4741 [21] onde é definido o protocolo NETCONF e mais três que definem o uso do NETCONF com diferentes protocolos de transporte: *Secure Shell* (SSH) [22], *Blocks Extensible Exchange Protocol* (BEEP)[23], *Simple Object Access Protocol* (SOAP)[24] e *Tansport Layer Security* (TLS) [25]. Entretanto, surgiram RFCs para definir a notificação de eventos [26], locks parciais [27] e um modulo YANG para monitorização [28]. Presentemente existe uma nova RFC para definir o protocolo NETCONF [29] e outra para a sua implementação sobre SSH [30], bem como uma RFC que define uma nova capaciade, *with-defaults* [31].

É incontestável que a gestão e monitorização têm um papel essencial na boa prestação de serviços de uma rede ou sistema. Podem evitar falhas no serviço ou ajudar na rápida recuperação dos mesmos. No entanto, o investimento económico e de recursos humanos nestas funcionalidades é normalmente menosprezado. As vantagens de uma boa gestão e monitorização já foi identificada hà muitos anos e conta com várias soluções que, seja pela sua complexidade ou por problemas relacionados com interoperabilidade, sempre apresentaram problemas para serem totalmente adoptados pelos administradores de redes ou sistemas. Por outro lado, as empresas não vêm esta área como potencialmente lucrativa, o que leva à falta de investimento e ao lançamento tardio dos modelos da dados dos seus equipamentos, muitas vezes num estado incompleto. Para colmatar estas falhas, as empresas e parceiros de investigação debateram o problema no sentido de encontrar soluções, resultando a iniciativa de criar um novo protocolo, o NETCONF, que tem em atenção as várias directivas do encontro, para um bom sistema de gestão e monitorização.

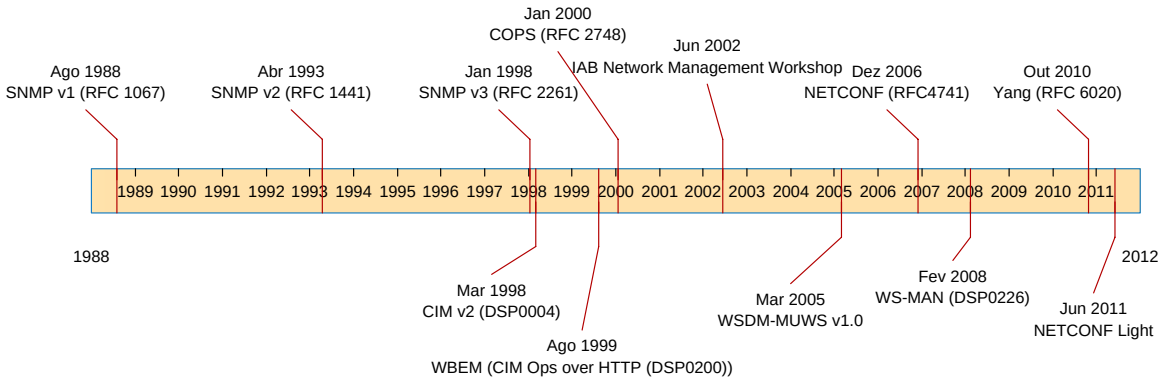

Figura 2. Cronograma dos eventos relevantes para esta dissertação

1.1. Estrutura da dissertação

Esta secção enquadra o contexto do aparecimento da tecnologia de gestão NETCONF, bem como a estrutura e objetivos da dissertação.

No segundo capítulo são analisados de forma detalhada diferentes tecnologias para gestão de redes. São descritas as suas características, funcionalidades e algumas limitações.

O terceiro capítulo descreve a implementação de NETCONF feita no decorrer deste trabalho, os requisitos cumpridos e dificuldades encontradas, os modelos de dados utilizados e as mensagens trocadas.

O quarto capítulo descreve um conjunto de testes efetuados à implementação NETCONF, bem como um conjunto de testes efetuados a implementações de tecnologias descritas no segundo capítulo. Descreve ainda uma análise comparativa dos resultados obtidos nos testes.

Por fim, são apresentadas as conclusões relativamente à implementação e aos resultados. São feitas algumas considerações em relação ao trabalho a ser desenvolvido futuramente.

1.2. Objetivos

Com este trabalho pretende-se apresentar uma solução de monitorização de redes baseada em NETCONF. As aplicações cliente e servidor usam SOAP (Simple Object Access Protocol) para transporte das mensagens RPC (Remote Procedure Call) e devem suportar as operações de monitorização através da notificação de eventos.

Posteriormente, pretende-se submeter a implementação do NETCONF e outras implementações dos protocolos que usam interfaces web na sua operação a testes de desempenho que permitam medir o tráfego gerado, a eficiência de sinalização e o tempo de resposta das suas capacidades de notificação de eventos.

No final, pretende-se comparar os resultados obtidos e concluir sobre as vantagens e desvantagens de cada aplicação. Pretende-se ainda avaliar a viabilidade do NETCONF para o futuro da gestão e monitorização de redes IP.

2. Tecnologias de Gestão de Rede

Neste capítulo analisam-se várias tecnologias para gestão de redes. Pretende-se ainda fazer algumas comparações em relação às capacidades e limitações de cada uma destas tecnologias, com especial relevo para as capacidades de monitorização.

Monitorização é o processo de obter informação acerca das configurações e estado dos vários elementos de um sistema computacional e preparar essa informação para consumo do sistema de gestão. A preparação da informação pode passar por criar relatórios para o administrador do sistema, recolher dados pertinentes de um conjunto de dados em bruto ou a compactação de um conjunto muito grande de informação para que esta possa ser interpretada por humanos. O tratamento adequado desta informação pode facilitar muito as tarefas de gestão, podendo a informação ser dividida em grupos lógicos relevantes, como:

- **Informação de estado** – informação acerca do estado de funcionamento de um dispositivo. Um dispositivo pode estar ligado e a funcionar corretamente, pode estar ligado mas com falhas de funcionamento ou estar desligado. Estas informações têm um impacto importante numa primeira análise durante as tarefas de gestão;
- **Informação das configurações** – A informação sobre as configurações consiste em todos os atributos de um dispositivo manipuláveis pelo administrador. A monitorização desta informação pode indicar se um atributo tem um valor válido, se é a causa para a degradação do desempenho do sistema ou mesmo responsável pela falha do mesmo;
- **Estatísticas de desempenho e utilização** - Estas informações são normalmente importantes para a gestão de recursos de um sistema. Aqui estão contempladas informações como o número de utilizadores ou sessões ativas, bem como, informações acerca da latência ou da taxa de transferência;
- **Informação de erros** – A informação de erros pode indicar falhas ou a operação incorreta de um dispositivo. Esta informação pode incluir códigos de erro ou o número de erros ocorridos num intervalo de tempo;
- **Informação da topologia** – Normalmente a topologia de uma rede já é conhecida quando se inicia o processo de monitorização. No entanto, conhecer atempadamente alterações da topologia pode ser vital para o bom funcionamento do sistema ou redução do seu tempo de inatividade ou mau funcionamento.

Estes conjuntos de informações não são estanques e devem ser adaptados aos dispositivos a que se destinam. Por exemplo, a informação do *throughput* de um servidor HTTP pode ser medida através do número de pedidos recebidos, num *router* pode ser pelo número de pacotes reencaminhados ou num servidor de *mail* pelo número de emails processados.

As aplicações de monitorização normalmente seguem uma estrutura de funcionamento idêntica à representada na **Figura 3**.

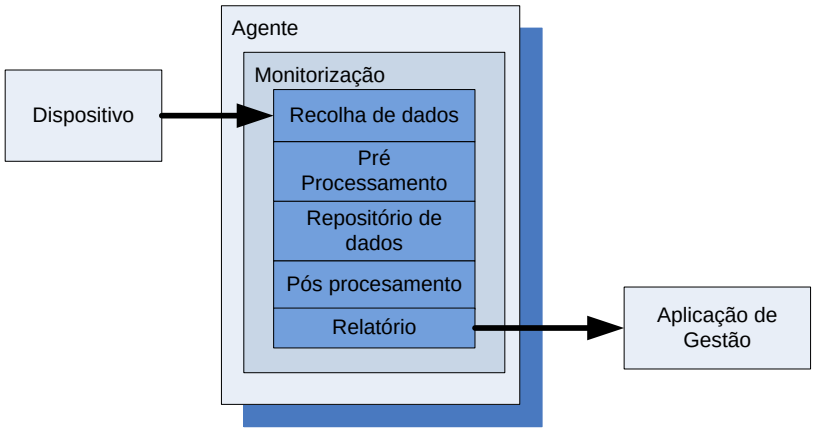

Figura 3. Estrutura genérica das aplicações de monitorização

A primeira componente do módulo de monitorização é a recolha de dados. O módulo recolhe dados em bruto dos dispositivos ou aplicações que constituem o sistema. Após a recolha de dados é necessário processá-los para armazenamento. Este processamento é feito em tempo real à medida que os dados são recolhidos e serve para verificar a integridade da informação, para eliminar dados redundantes ou desnecessários e para efetuar as conversões necessárias entre os tipos de dados recolhidos e os tipos de dados do repositório. O armazenamento mantém a persistência dos dados para análise e posterior geração de relatórios. Esta persistência é necessária e permite verificar a evolução dos dados ao longo do tempo e análises de desempenho difíceis de fazer em tempo real. Para gerar os relatórios é necessário processar os dados armazenados; a qualidade da informação de um sistema de gestão é extremamente dependente dos algoritmos utilizados para processar os dados nesta fase. Os relatórios são uma mera apresentação ao administrador dos resultados do pós-processamento.

A recolha de dados de monitorização pode ser dividida em dois modos, monitorização passiva e monitorização ativa. A monitorização passiva é feita através das informações recolhidas pelo normal funcionamento do sistema. Não implica nenhum esforço adicional para o sistema, a não ser o envio da informação para a consola de gestão. A monitorização passiva utiliza *logs*, configurações, MIBs, *sniffers* de redes, entre outros, para produzir informação. O uso de notificações também é considerado monitorização passiva. Por outro lado, a monitorização ativa interfere com a carga de trabalho (*workload*) do sistema. Implica pedidos explícitos para determinar a informação sobre o sistema. Por exemplo, para medir a latência de um servidor web uma aplicação pode periodicamente consultar um *site* para analisar a latência ou *uptime*. Outros exemplos são a utilização de ferramentas como o *ping* ou o *traceroute*, bastante utilizados em monitorização.

As notificações de eventos são um método muito utilizado para fiscalização do funcionamento de um sistema. Neste método participam entidades gestoras que subscrevem eventos e participam agentes que recebem as subscrições e ocasionalmente enviam as notificações necessárias. Os eventos são acontecimentos de interesse, como alterações nas configurações ou mudanças de estado. Neste capítulo descreve-se como é que as atuais tecnologias de rede implementam estes sistemas de notificações e quais as suas características.

2.1. SNMP

O *Simple Network Managment Protocol* (SNMP) é um protocolo da camada aplicacional, desenvolvido pelo *Internet Architecture Board* (IAB) no fim dos anos 80. A RFC 1157 [8] define o SNMP para gestão e configuração de redes IP. É o protocolo mais utilizado e difundido para Sistemas de Gestão de Redes (SGR).

O SNMP usa uma arquitetura *manager-agent* [9], ilustrada na **Figura 4**. As suas operações baseiam-se em interações entre três componentes básicos:

- Agentes – são aplicações que correm nos dispositivos que se pretendem gerir e são responsáveis por interpretar os pedidos do gestor e consultar ou manipular as MIBs no sentido de os satisfazer. Pode, quando devidamente solicitado, enviar informação de forma assíncrona para o gestor;
- Gestores – comunicam com os agentes para pedir informação acerca das configurações e estado dos dispositivos;
- MIB – são tabelas que representam os modelos de dados dos dispositivos.

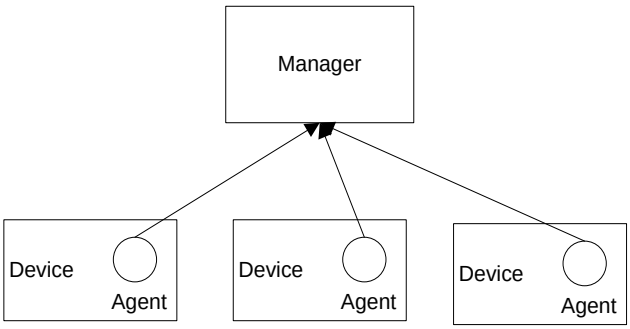

Figura 4. Esquema da arquitetura *manager-agent*

Para representação da informação de gestão o SNMP usa a notação *Structure of Management Information* (SMI) [32], que se baseia no *Abstract Syntax Notation One* (ASN.1) [33]. O SMI é usado para definir *Managment Information Bases* (MIB) [34] que representam os modelos de dados presentes nos diferentes dispositivos. É possível representar os dados de duas maneiras: a escalar e a tabular. A forma escalar representa uma instância de um objeto, enquanto a forma tabular permite representar e relacionar múltiplas instâncias de objetos que são agrupados nas tabelas MIB. As entradas das tabelas são identificadas por *Object Identifiers* (OID) que identificam de forma hierárquica o caminho até um nó. Por exemplo, o valor "*CPU Load*" encontra-se em: **iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).host(25).hrDevice(3). hrProcessorTable(3).hrProcessorEntry(1).hrProcessorLoad(2)** da hierarquia de OIDs (**Figura 5**). Para simplificar, este elemento pode ser representado apenas por

1.3.6.1.2.1.25.3.3.1.2. As MIBs associam a cada OID um valor escalar. A falta de suporte para definição de estruturas de dados, operações e objetos [9], [2] retiram ao SMI a flexibilidade necessária para lidar com as necessidades atuais da gestão de redes.

No que se refere à interação entre agentes e gestores, o SNMP define um conjunto de operações para visualização e manipulação dos dados de configuração:

- *GetRequest*: Usado para ler uma entrada de uma MIB de um agente;
- *Getnextrequest*: Usa-se quando se quer devolver uma lista ou tabela de objetos. Inicialmente envia-se um *Getrequest* e depois uma sequência de *Getnextrequest*. Este método pode implicar um grande número de pedidos o que leva a uma má utilização da largura de banda [2];
- *Getbulkrequest*: Versão otimizada do *Getnextrequest* que surgiu no SNMPv2 [35] e permite pedir vários objetos de uma só vez;
- *Setrequest*: permite alterar o valor de um objeto;
- *Response*: usado pelos agentes para responder aos pedidos de informação e alteração das MIBs;
- *Trap*: é uma notificação assíncrona enviada por um agente para um gestor, na sequência de algum evento definido previamente pelo gestor;
- *Inform*: semelhante ao *Trap* mas implica que o recetor envie uma resposta para confirmar a receção da notificação. Esta mensagem é usada para notificações entre gestores e é necessária uma vez que o SNMP corre sobre UDP, que não garante a receção de mensagens.

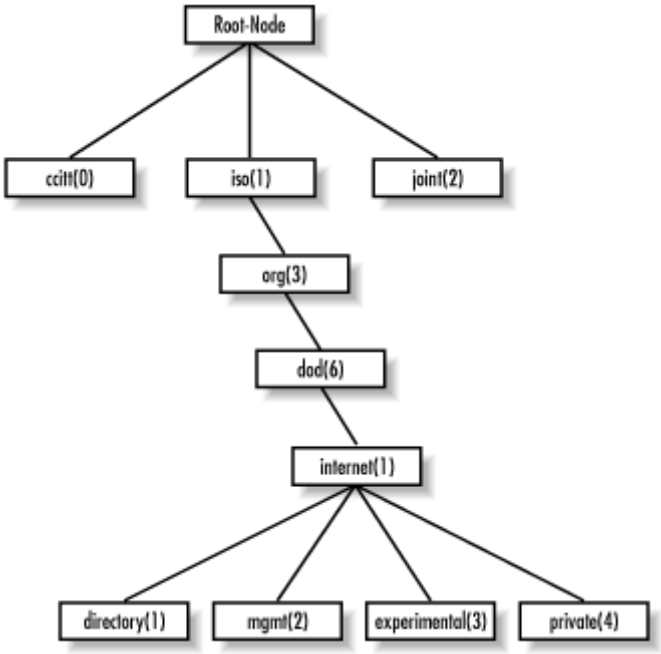

Figura 5. Árvore de Objetos do SMI [36]

A subscrição de notificações no SNMP é feita através da alteração do valor da restrição de acesso definido na tabela de controlo de acessos, definido na "Party MIB" [37]. Um gestor deve através de um *SetRequest* definir o valor *aclPrivileges* de forma a integrar a operação de *Trap* ou *Inform*. Assim, dão origem a notificações todas as entradas desta tabela que tiverem este valor configurado para *traps* ou *informs*, bem como o valor que indica o destino da notificação e o que descrimina quais os recursos que dão origem às notificações. Então, quando ocorrem eventos suscetíveis de gerar uma notificação, é enviado um *Trap* ou um *Inform* para o gestor que se encarrega de o interpretar.

Na sua primeira versão, o SNMPv1[8] só definia o *GetRequest*, o *GetNexRequest*, o *SetRequest* e o *Trap*. A operação *GetRequest* apenas permitia ler uma entrada de uma MIB, o que implicava que para ler várias entradas ou mesmo uma configuração completa houvesse uma grande troca de mensagens, o que consumia muita largura de banda. Como tal, surgiu a segunda versão, SNMPv2 [35], onde foram definidas as operações *GetBulkRequest* para leitura de várias entradas numa tabela MIB e *Inform* para transferência síncrona de notificações de eventos. Nestas duas versões a segurança era implementada através da "*comunity name"* que era transmitida em formato de texto, que podia ser facilmente visualizada através da inspeção do tráfego da rede. A fraca segurança do SNMP limitou o seu uso às funções de monitorização, uma vez que, os administradores de rede não arriscavam operações mais "invasivas" como alterar parâmetros das MIBs. Posteriormente foi definida a versão três do protocolo, SNMPv3 [38], que implementava autenticação de agentes e gestores, cifra das comunicações e controlo de acesso.

Embora tenha sido o principal protocolo para monitorização de redes [2], o SNMP não respondia de forma aceitável às necessidades das redes de comunicação atuais. Este facto deve-se a vários fatores: a falta de flexibilidade do SMI para descrever dados complexos; a falta de operações de *lock* que permitissem a vários gestores comunicar em simultâneo com um agente mantendo a consistência, tanto dos dados no agente como das leituras; a impossibilidade de distinguir dados de configuração e de estado [39]; a extrema complexidade para ler, operar e escrever configurações completas; a parca flexibilidade para gestão de vários dispositivos ou a escolha do UDP como protocolo de transporte que não garante a fiabilidade das comunicações. Chegou a existir uma proposta para normalizar o transporte sobre TCP [40], mas nunca passou do estado de experimental. As mensagens do UDP são limitadas pelo *Maximum Transfer Unit* (MTU), o que pode criar problemas quando se usa a operação *GetBulkRequest*, cuja resposta pode facilmente ultrapassar o limite definido[41]. Para lidar com este problema podiam-se utilizar protocolos orientados à ligação, como no WBEM ou no NETCONF que funcionam sobre TCP, o qual, através do controlo de fluxos e retransmissão de pacotes garante a fiabilidade das comunicações.

O SNMP sempre demonstrou ser muito complexo e pouco flexível [2], [18], [41], e os mecanismos de gestão de credenciais para autenticação que estão previstos no SNMPv3 são suportados por muito poucos equipamentos [42] e apresentam um elevado grau de complexidade [11]. Apesar dos vários ciclos de aperfeiçoamento, a utilização do SNMP sempre ficou reduzida às funções de monitorização [2], [9], [18].

2.2. Tecnologias desenvolvidas pelo *Distributed Managment Task Force* (DMTF)

O DMTF[43] é uma organização constituída por várias empresas, como a AMD, Cisco, Citrix, EMC, HP, Huawei, Intel, Microsoft, IBM, Oracle, VMWare, entre outras, que pretendem desenvolver, testar, promover e adotar normas de sistemas de gestão distribuída. O objetivo desta organização é encontrar soluções interoperáveis e independentes dos fabricantes ou plataformas. Esta organização tem uma larga experiência no desenvolvimento de normas para este tipo de soluções, tendo por exemplo normas para gestão de ambientes virtuais, bases de dados ou para gestão de *Clouds*. Assim, o DMTF conta com uma vasta gama de normas para gestão (**Figura 6**)**.** São exemplo das tecnologias do DMTF o *Web Based Enterprise Management* (WBEM)[44] e o *Web Services for Management* (WS-MAN)[45] que vão ser explorados nas secções seguintes.

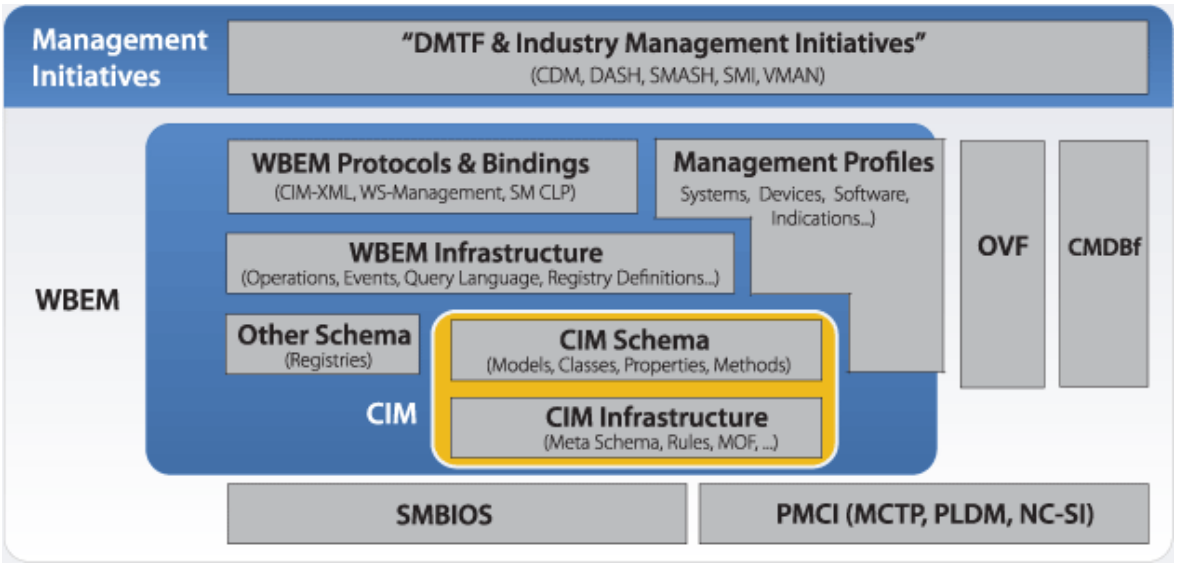

Figura 6. Diagrama das tecnologias do DMTF [46]

2.2.1. WBEM

O *Web-Based Enterprise Managment* (WBEM) é um conjunto de normas para unificar a gestão distribuída de ambientes computacionais, multiplataforma e de diferentes fabricantes.

O WBEM baseia-se em três normas base:

- ✓ *Common Information Model* (CIM) [13] [47], para representar dados;
- ✓ CIM-XML [48], para codificar dados;
- ✓ CIM-operations-over-HTTP [49], para a transferência de dados.

O CIM é uma linguagem orientada a objetos, para representar informação de gestão. Usa o *Unified Modeling Language* (UML) para fazer uma descrição conceptual da informação e recorre ao *Management Object Format* (MOF) para a representação formal dos dados. A **Figura 7** ilustra a relação entre a linguagem de modelação e a linguagem de definição de dados.

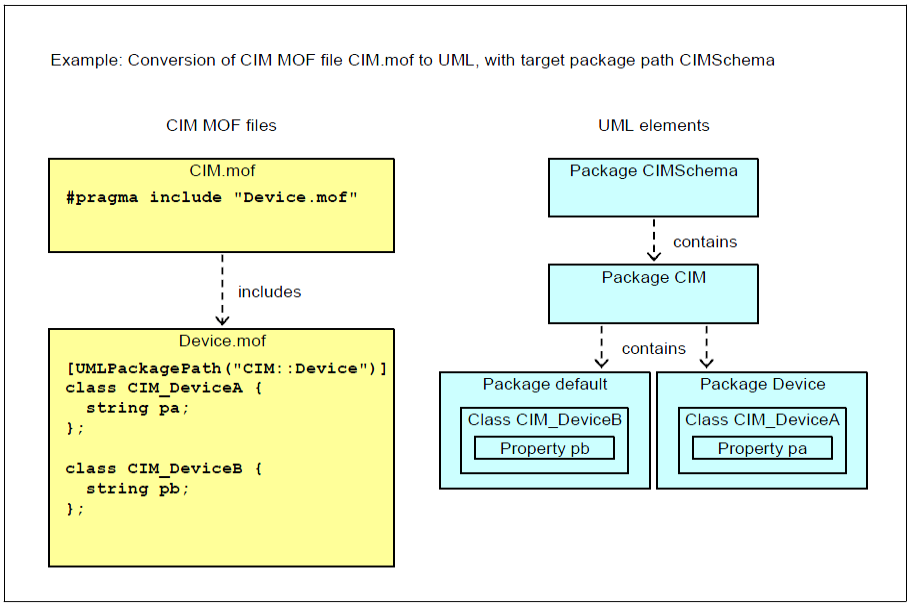

Figura 7. Exemplo da conversão de MOF para UML [50]

O CIM divide-se em três camadas, constituídas por conjuntos de classes: o *core model*, o *common model* e *extensions schemas*. O *core model* diz respeito às definições comuns a todas as áreas de gestão, é um pequeno conjunto de classes, associações e propriedades para analisar e descrever sistemas de gestão. O *common model* é um conjunto de classes que define áreas abrangentes como sistemas, redes, equipamentos ou aplicações, independentemente de tecnologias ou implementações. Os modelos *core* e *common* são normalmente referidos como CIM *schemas* e constituem uma base programática para definir aplicações de gestão. Adicionalmente surgem modelos específicos dos recursos, os

extension schemas, que podem por exemplo ser definições para servidores, desktops, periféricos, sistemas operativos, aplicações ou equipamentos de rede.

O CIM-XML define uma gramática XML, escrita em *Document type definition* (DTD), para representar as classes e instâncias do CIM bem como as operações CIM definidas sobre HTTP.

A norma CIM-operations-over-HTTP mapeia as mensagens CIM para HTTP para que as implementações do CIM possam funcionar de forma normalizada com aplicações HTTP.

O WBEM segue uma arquitetura cliente-servidor, como representado na **Figura** 8. O cliente do WBEM codifica e envia os pedidos em XML para o servidor. Enquanto, o servidor do WBEM gere os objetos de gestão e responde aos pedidos dos clientes.

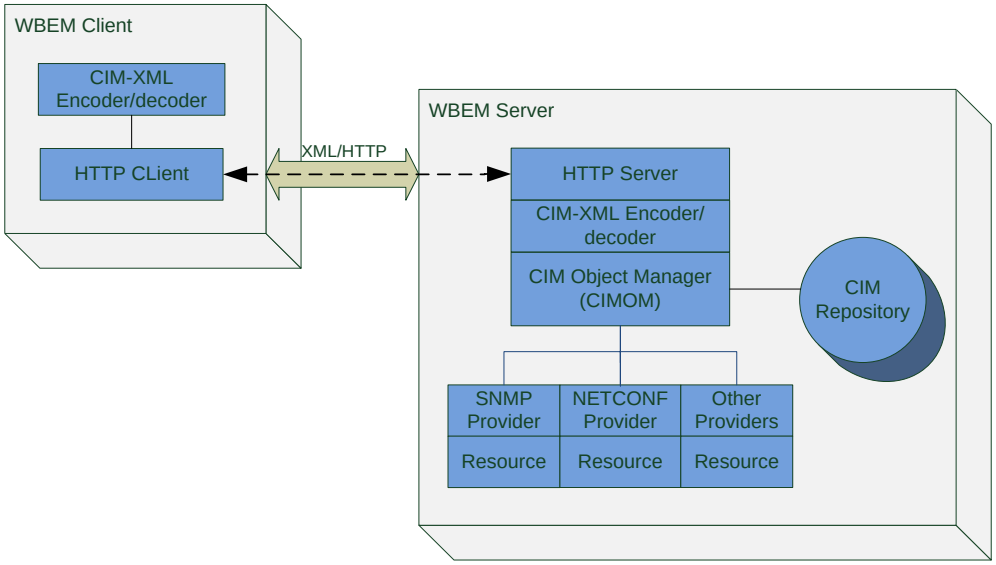

Figura 8. Arquitetura de um sistema WBEM

As mensagens são recebidas pelo servidor HTTP, responsável pelas comunicações e são descodificadas pelo CIM-XML *Decoder*. Posteriormente, são entregues ao *CIM Object Manager* (CIMOM) que interpreta os pedidos do cliente de WBEM e consulta o *CIM Repository* pela informação pretendida; caso esta não esteja disponível, consulta um *provider* que corresponda à operação CIM recebida. Os *providers* comunicam diretamente com os recursos para obter os dados pretendidos e reencaminham-nos para o CIMOM.

Numa visão mais abrangente do sistema de gestão, **Figura 9**, um cliente pode ser visto como um gestor de uma arquitetura *manager-agent* e o servidor como um agente. Neste caso, o cliente WBEM é a *interface* da arquitetura WBEM que comunica com uma aplicação de gestão fornecendo uma interação *web-based* com o utilizador. Esta *interface*

permite fazer pedidos ao gestor que os remete para os respetivos agentes e permite receber as notificações passadas pelo gestor.

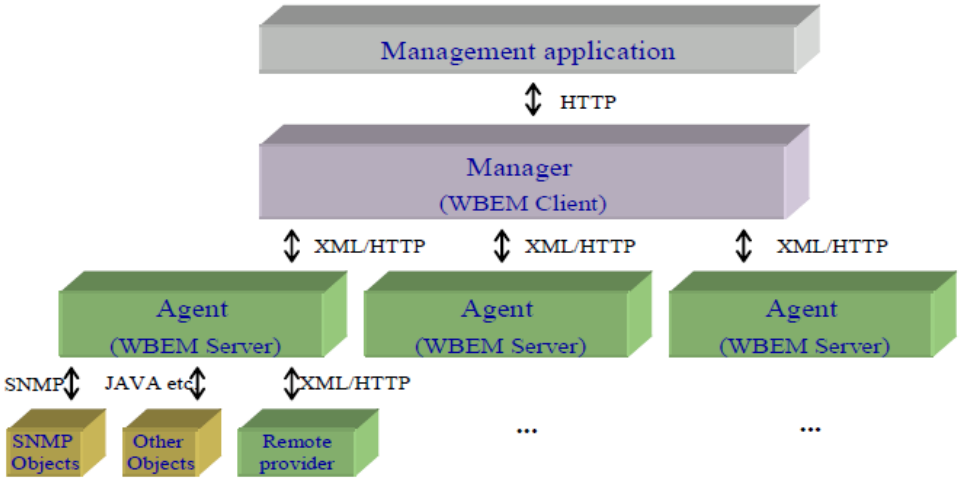

Figura 9. Arquitetura de um sistema de gestão [51]

Para implementar as funcionalidades de monitorização o WBEM implementa *Indications* como sendo mensagens que notificam sobre a ocorrência de eventos. As *Indications* podem ser de dois tipos: *"Life Cycle Indications"* referentes a alterações nas classes e instâncias CIM ou *"Process Indications"* referentes a objetos que não são obrigatoriamente modelados em CIM, como por exemplo, as *traps* do SNMP. A subscrição de *Indications* faz-se através da instanciação da classe *CIM_IndicationSubscription* que associa uma instância da classe *CIM_IndicationFilter* e outra da classe *CIM_ListenerDestination*, visível no diagrama de classes da **Figura 10**. A classe de filtragem permite, através da linguagem *WBEM Query Language* (WQL) selecionar os dados, alvo da *Indication*. Já a classe *ListenerDestination* indica o destino da *Indication* e permite definir o protocolo e codificação de transporte. Uma subclasse frequentemente utilizada é a *CIM_ListenerDestinationCIMXML* que codifica as *Indications* em CIM/XML e usa HTTP para transporte, mas podem ser definidas outras, por exemplo para usar protocolos de *multicast*, de correio eletrónico ou de *messaging* para transporte das *Indications*.

Uma vez que o WBEM funciona sobre HTTP é possível usar SSL/TLS para segurança das comunicações, ou seja, usar HTTPS. Para reforçar a segurança o CIM-XML implementa mecanismos de autenticação, permitindo ao cliente enviar login e palavra-chave para o servidor, bem como mecanismos para gestão de permissões normalmente definidas por *namespace*.

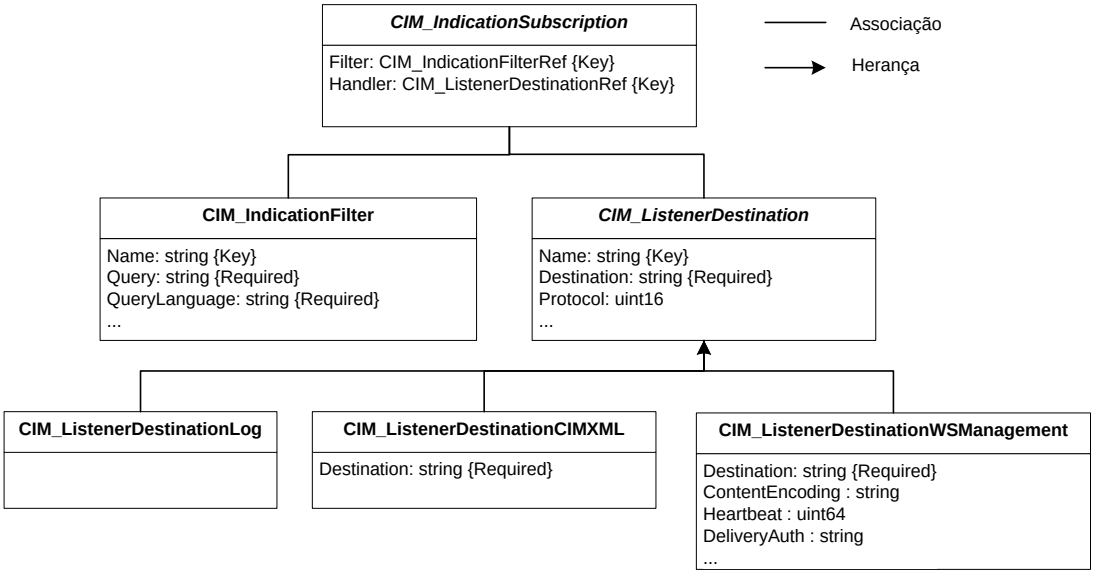

Figura 10. Diagrama de Classes de *Indications* do WBEM

O WBEM conta com várias soluções proprietárias e nos últimos anos têm surgido vários projetos *open-source*, alguns com o apoio de empresas de renome. A HP fornece um pacote de serviços baseado no WBEM, o *HP WBEM Solutions*, que inclui o *HP WBEM Services*, o *HP WBEM Providers*, o *HP WBEM Clients* e o *HP WBEM SDK* [52]. A Oracle também fornece uma implementação das normas WBEM e CIM para o seu sistema operativo, o Solaris. O *Solaris WBEM Services* inclui um gestor de objetos CIM, um compilador MOF, Solaris schema e *Solaris WBEM SDK* [53]. A Cisco inclui em vários dos seus equipamentos agentes/servidores compatíveis com as normas WBEM e CIM [54]. Também a Microsoft, integra no seu sistema operativo MS Windows o *Windows Management Instrumentation* (WMI) [55], que é a sua implementação do WBEM. Esta implementação tem um agente/servidor, *WMI service*, e repositórios CIM, *WMI repository* e disponibiliza vários *providers* relacionados com os seus produtos, como por exemplo o *Boot Configuration Data* (BCD) *provider* que garante acesso a dados relacionados com a inicialização do SO através das classes do BCD [56]. Existe ainda a WBEM Solutions [57], que desenvolve, licencia e suporta soluções baseadas em tecnologias de gestão de redes como WBEM e o CIM. Na sua gama de produtos, a *WBEM solutions* disponibiliza um SDK para criar aplicações cliente e servidor para aplicação em meio empresarial. Nas aplicações de gestão existem implementações da IBM, o *IBM Director* [58] e o *CIM Solutions* [59] da DELL que conta com um *CIM browser* e um *CIM provider*.

No segmento de implementações *open-source* existe o *WBEM Services Project* [60], que é uma implementação de uma API em java, independente do sistema operativo, que inclui uma aplicação servidor, uma cliente e ferramentas WBEM. O *OpenPegasus* [61] é uma implementação em C++, cujos objetivos principais são a portabilidade e modularidade. Esta implementação tem atualmente versões para Unix, Linux, OpenVMS e Microsoft Windows. Outro projeto de referência é o *OpenWBEM* [62] também este, escrito

em C++ e implementa um agente/servidor WBEM e uma framework que permite integrar funcionalidades de gestão de configurações, monitorização de desempenho e uma solução empresarial de gestão. O *Standards Based Linux Instrumentation* (SBLIM) [63] é um projeto iniciado pela IBM para desenvolver ferramentas de gestão para sistemas Linux utilizando as diretivas da iniciativa WBEM. O SBLIM implementa clientes WBEM como o *wbemcli* ou o *Java CIM Client*, aplicações servidor como o *Small Footprint CIM Broker* (SFCB) [64] e vários *providers* para o ambiente Linux, por exemplo existem *providers* para verificar informação sobre os processos do SO ou sobre os dispositivos acessíveis pelo *kernel*. Também no ambiente Linux, existe o *KDE CIM Browser* [65] que é uma aplicação para gestão de agentes/servidores WBEM. Nas aplicações de gestão existe também o *CIM Navigator* [66], o *JavaCIM client* [67] do projeto *OpenPegasus*, o *Yet Another WBEM Navigator* (YAWN) [68] ou o *ECUTE* [69], que para além do *ECUTE Explorer*, para visualizar dados exportados de um CIMOM, oferece o *ECUTE Modeler*, para modelar classes CIM em UML e exportá-las para MOF, e o *ECUTE Analyzer* para analisar o tráfego entre um cliente e servidor WBEM.

O WBEM tem origem numa iniciativa do DMTF que visa facilitar a vida dos administradores de sistemas e reduzir os custos de manutenção, agilizando a gestão remota de sistemas computacionais. É principalmente usado em infraestruturas de grande dimensão e complexidade onde dominam as soluções fornecidas pelas grandes empresas da área. Assim sendo, o WBEM não tem tão boa implantação de mercado em negócios mais pequenos e alguns projetos *open-source*, como o WBEMSource Initiative [70], foram cancelados e produtos comerciais como o Purgos [71] foram descontinuados. No entanto, o WBEM oferece um modelo de dados suficientemente flexível para representar dados com um alto nível de complexidade. Facilita o desenvolvimento de aplicações fornecendo classes base com definições comuns de gestão para sistemas, redes, aplicações e serviços, associando também uma boa integração com tecnologias já existentes. Atualmente, o WBEM é uma ferramenta a ter em conta para a gestão flexível, remota e distribuída de sistemas.

2.2.2. Web services para gestão

Web Services são uma tecnologia desenvolvida pelo World Wide Web Consortium (W3C) [72] e é uma plataforma de software concebida para suportar interações entre máquinas numa rede, promovendo interoperabilidade, redução da complexidade de implementação e facilidade na transposição de *firewalls*. A interface de um *web service* é definida usando *Web Service Defenition Language* (WSDL) [73] e tipicamente a troca de mensagens é feita recorrendo a SOAP encapsulado em mensagens HTTP, embora existam outras possibilidades, tal como é ilustrado na **Figura 11**.

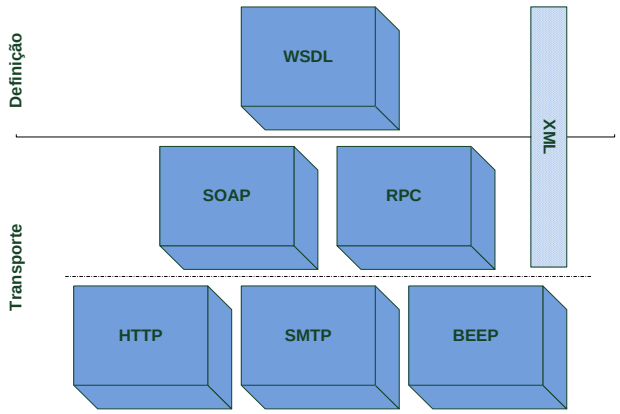

Figura 11. Estrutura em camadas de um *Web service*

Inicialmente, aquando da adoção de *web-services* para gestão, mais uma vez, houve muitas dúvidas acerca do impacto que a utilização do XML, pela sua verbosidade, poderia ter no desempenho das operações de gestão tradicionalmente efetuadas pelo SNMP que usa mensagens mais compactas. No entanto, vários estudos concluíram que o desempenho dos *web-services* era aceitável para as operações de gestão [74]. Como tal, existiram vários esforços da indústria para desenvolver normas para gestão usando *web services*. Desses esforços, tiveram impacto duas normas: uma da *Organisation for the Advancement of Structured Information Standards* (OASIS) [75], o *Management Using Web Services* (MUWS) [76][77]; e outra do *Distributed Management Task Force* (DMTF), o *Web Services for Management* (WS-MAN) [78].

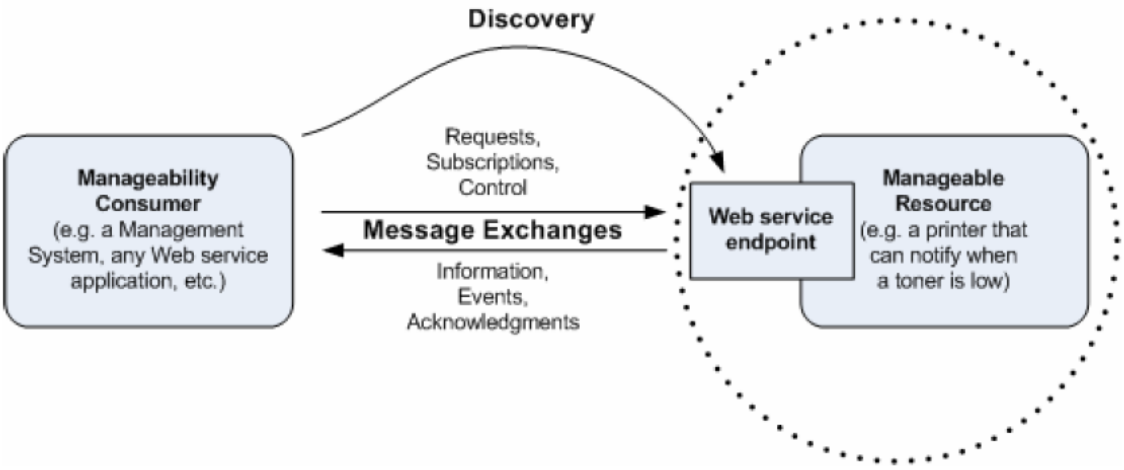

Figura 12. Arquitetura do WSDM [76]

A OASIS é um consórcio sem fins lucrativos, constituído por empresas como a Microsoft, a IBM, a Alcatel-Lucent, a Adobe, a AOL, a HP, a Oracle, entre muitas outras num universo de 600 empresas. A OASIS preocupa-se com o desenvolvimento de soluções globais e interoperáveis para áreas como a segurança, *Cloud Computing*, *Service Oriented Architechture* (SOA), *web services*, *Smart Grids*, gestão para emergências, entre outras. Dessas soluções, faz parte a norma Web Services Distributed Management (WSDM) que é uma iniciativa para definir uma arquitetura (**Figura 12**) para gerir recursos usando *web services*. A norma é constituída por duas especificações distintas, o Management Using Web Services (MUWS) e o Management of Web Services (MOWS) [79].

O MUWS define os mecanismos básicos e *Message Exchange Patterns* (MEPs) (**Tabela 2**) para gestão local ou remota de recursos usando *web-service*s, enquanto o MOWS se preocupa com a gestão de *web-service*s e pode ser visto como um caso particular do MUWS em que os recursos são os próprios *web service*s [80]. Neste trabalho foca-se o MUWS, uma vez que é a norma que se preocupa com as funcionalidades comparáveis ao SNMP e ao WS-MAN do DMTF.

Esta norma especifica como é que a gestão de recursos se torna possível usando *web-services*. Para isso, o MUWS tira partido de várias normas já existentes, descritas na **Tabela** 1, para desempenhar as suas funções.

Tabela 1. Normas utilizadas pelo MUWS

Norma	*Descrição*
SOAP [81]	Protocolo para a transferência de mensagens entre *web service*s baseado em XML
XML schema [82]	Descrição do conteúdo de documentos XML
WSDL [73]	Linguagem para definir *web service*s
WS-Addressing [83]	Define um esquema de endereçamento para *web service*s
WS-Resources [84]	Define as relações entre os *web service*s e os recursos
WS-ResourceProperties [85]	Define as propriedades dos recursos, bem como as operações que permitem manipular o valor das propriedades
WS-ResourceLifeTime [86]	Define operações para remover recursos e propriedades que permitam monitorizar o tempo de vida de um recurso
WS-ServiceGroup [87]	Define a criação de grupos de recursos
WS-BaseFaults [88]	Define uma normalização de mensagens de erro
WS-BaseNotification [89]	Define a estrutura base das mensagens de subscrição e notificação
WS-BrokeredNotification [90]	Prevê a distribuição de notificações através de intermediários
WS-Topics [91]	Define a representação XML dos tópicos das subscrições de eventos
WS-Security [92]	Prevê funcionalidades para garantir a integridade e confidencialidade na troca de mensagens SOAP

A **Figura 12**, mostra uma abordagem conceitual do WSDM. Os *web service enpoints* dão acesso aos *manageable resources*. Os *manageability consumers* descobrem os *web services endpoints* e trocam mensagens para pedir informação, subscrever eventos ou controlar o *manageable resource* associado ao endpoint. Para descobrir *web services*

endpoints, o *manageable consumer* deve obter um *endpoint reference* (EPR), como definido na norma WS-Addressing e representado na **Figura 13**. O MUWS usa este mecanismo, que é usado pela maioria das implementações de *web service*s para garantir a interoperabilidade entre as diferentes aplicações. A um *web service endpoint* que dê acesso a um *manageable resource* é-lhe atribuída a denominação de *manageability endpoint*. Assim, um *manageability consumer* utiliza a informação contida num EPR, por exemplo endereço e referências para propriedades, para interagir com um *manageability endpoint*. Estão previstos três tipos de interação, suportados pelos MEPs:

- O *manageability consumer* pode pedir/ler informação sobre as propriedades de um *manageable resource*,
- O *manageability consumer* pode manipular a informação das propriedades de um *manageable resource*,
- Um *manageable resource* pode notificar o *manageability consumer* de eventos que tenham sido previamente subscritos.

```
1  <wsa:EndpointReference>
2      <wsa:Address>xs:anyURI</wsa:Address>
3      <wsa:ReferenceProperties>... </wsa:ReferenceProperties>
4      <wsa:ReferenceParameters>... </wsa:ReferenceParameters>
5      <wsa:PortType>xs:QName</wsa:PortType>
6      <wsa:ServiceName PortName="xs:NCName">xs:QName</wsa:ServiceName>
7      <wsp:Policy> ... </wsp:Policy>*
8  </wsa:EndpointReference>
```

Figura 13. Exemplo da representação XML de um EPR

Na seguinte tabela constam os MEPs disponibilizados para as interações entre o *manageability consumers* e *manageability resources*.

Tabela 2. WSDM Message Exchange Patterns

Pedir informação sobre as propriedades de um recurso	
GetResourceProperty	Devolve o valor de uma propriedade de um recurso
GetMultipleResourceProperties	Devolve os valores de várias propriedades de um recurso
QueryResourceProperties	Permite ler uma parte de uma propriedade de um *manageable resource* usando uma linguagem como o XPath [93]
QueryRelationshipsByType	Devolve informação sobre uma relação em que o recurso participa. (relações definem associações entre recursos)
Manipular as propriedades de um Recurso	
SetResourceProperties	Permite inserir, atualiza ou apagar propriedades de um recurso
Subscrições e Notificações	
Subscribe	Subscrever notificações
GetCurrentMessage	Pedido da última notificação de um tópico
PauseSubscription	Pausar temporariamente a notificação de eventos para um *manageability consumer*
ResumeSubscription	Retomar o envio de notificações referentes a um tópico
Notify	Permite receber notificações a partir de um *manageable consumer*
RegisterPublisher	Regista num NotificationBroker, um *manageable resource* como um emissor de notificações
Destroy	Apaga o registo de um *manageable resource* num NotificationBroker (recipiente/distribuidor de notificações)

O WS-MAN, é uma norma desenvolvida pelo subgrupo *WS-Management* do *WBEM Infrastructure & Protocols Working Group* para, de forma remota e usando *web services*, gerir equipamento ou serviços presentes numa rede de informação. Esta norma promove a interação entre *web service*s baseada em *Simple Object Access Protocol* (SOAP) [81]. Estes *web-services* tanto implementam aplicações de gestão como os objetos que se pretendem gerir. Como noutras tecnologias de gestão do DMTF, o CIM é a norma escolhida para descrição dos dados de gestão e operações. O WS-MAN apresenta a estrutura protocolar ilustrada na **Figura 14**.

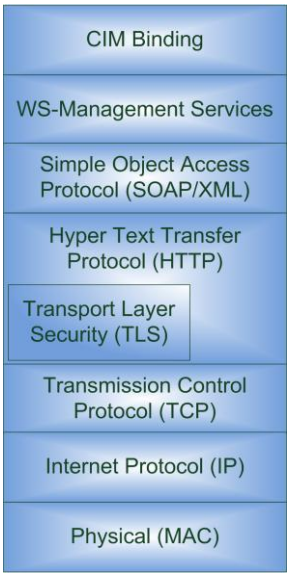

Figura 14. Stack protocolar do WS-MAN

O WS-MAN foi pensado para a gestão dos sistemas distribuídos tradicionais e os mais recentes sistemas baseados em *Service Oriented Architecture* (SOA). Esta arquitetura estrutura os serviços de forma modular utilizando *web services*. Permite a reutilização desses módulos para otimizar e agilizar a conceção e implementação de novos serviços. O WS-MAN tira partido dessa filosofia e aproveita várias normas existentes ou algumas implementadas para o efeito, descriminadas na **Tabela 3**, para normalizar a gestão usando *web service*s.

A norma WS-Management define um conjunto base de operações que devem ser disponibilizadas e que devem ser comuns a todos os *web service*s de gestão:

- Descobrir os elementos de gestão disponíveis e navegar entre eles,
- Ler, escrever, criar e apagar recursos individuais de cada elemento,
- Ler grandes conjuntos de informação, como tabelas ou logs,
- Subscrever eventos gerados pelos elementos geridos,
- Executar operações específicas de gestão com input e output bem definidos.

A norma define os requisitos mínimos a implementar para haver compatibilidade entre as diferentes implementações dos *web service*s. O WS-MAN permite a extensão das funcionalidades mencionadas mas não obriga à implementação de todas as funcionalidades, caso por exemplo não se adequem ao sistema em questão.

Para garantir compatibilidade entre *web service*s e permitir acesso remoto, o WS-MAN indica dois modos de endereçamento: o endereçamento definido pelo WS-Addressing e um mecanismo definido pelo WS-MAN. Permite ainda usar outros modelos não especificados na norma, desde que se mostrem adequados. Os modelos usam *endpoint references* (EPR) que descriminam a informação necessária para aceder a um web-service. O modelo da norma, embora não seja obrigatório, é tido como o modelo que oferece melhores condições para interações bem sucedidas entre *web service*s.

Tabela 3. Normas utilizadas no WS-Management

Norma	*Descrição*
SOAP	Protocolo para a transferência de mensagens entre *web service*s baseado em XML
WSDL	Linguagem para definir *web service*s
XML schema	Descrição do conteúdo de documentos XML
WS-Transfer	Operações de acesso aos recursos de gestão
WS-Policy	Define modelos e sintaxe para descrever políticas de um serviço
WS-Trust	Cobre a emissão, troca e validação de tokens para segurança
WS-Enumeration	Define operações para transmissão de listas de elementos XML
WS-Eventing	Define operações para criar, gerir e cancelar subscrições de eventos
WS-Addressing	Define um esquema de endereçamento para *web service*s
WS-Security	Prevê funcionalidades para garantir a integridade e confidencialidade de mensagens SOAP

Este modelo representa um EPR através de um conjunto de parâmetros baseados em campos de um cabeçalho SOAP:

- wsa:To – endereço do web-service (elemento obrigatório);
- wsman:ResourceURI – identificador que representa uma classe de recursos (elemento obrigatório);
- wsman:SelectorSet – identificador de uma instância específica, caso existam várias instâncias da mesma classe ou recurso (opcional).

O WS-MAN define o mapeamento que deve ser feito entre um EPR e o cabeçalho SOAP. No entanto, podem ser definidos outros mapeamentos recorrendo a *Web services Definition Language* (WSDL) [73], embora, para garantir compatibilidade entre implementações o mapeamento SOAP deva ser sempre suportado. Para melhor interação entre *web service*s, o WS-MAN permite "*nested* EPRs", que são EPRs encapsulados na tag SelectorSet dentro de outro EPR. Um exemplo prático desta funcionalidade é um web-

service que forneça dados estatísticos sobre outros *web-service*s. Na *tag SelectorSet* descriminam-se *Selectors* que identificam uma instância de um recurso e são constituídos pelo identificador do recurso e por um valor. Para filtragem podem-se combinar vários Selectors através de ANDs lógicos.

Como já foi mencionado, o WS-MAN herda operações de outras normas para promover a interoperabilidade entre *web service*s. As funcionalidades do WS-MAN são garantidas por quatro operações básicas:

- *Get* – devolve o valor de um recurso,
- *Put* – atualiza o valor de um recurso,
- *Delete* – apaga o recurso,
- *Create* – criar uma nova instância de um recurso.

Para operar sobre conjuntos de recursos deve ser definida a operação iterativa *Enumerate,* que inicia a enumeração de itens de um recurso. Nesta operação é possível criar filtros que podem ser definidos por predicados booleanos baseados em *selectors, Structured Query Language* (SQL)[94] ou *CIM Query Language* (CQL)[95]. Esta funcionalidade não é obrigatória, como tal a sua inexistência elimina a possibilidade de filtragem de dados prevista pelo WS-MAN, o que prejudica gravemente a funcionalidade de uma implementação sem esta capacidade. No contexto de uma enumeração é possível percorrer os itens um a um usando a operação *Pull,* que ao mesmo tempo retira esses itens da fila de enumeração. Estes contextos de enumeração terminam quando se chega ao fim da fila, ou com a execução da operação *Release* que liberta os recursos alocados para a fila. Para saber o estado de um contexto de enumeração existe a operação *GetStatus*, cuja resposta indica se o contexto ainda está ativo e por quanto tempo. Consequentemente é possível usar a operação *Renew,* para renovar o período de um contexto de enumeração.

Para a monitorização o WS-MAN usa a norma WS-Eventing [96], que define um conjunto de operações que permitem a um *web service* com funções de gestão registar o seu interesse em receber notificações de eventos (*Subscribe)*. Estas subscrições têm tempo de validade, findo o qual a fonte de eventos cessa o envio de notificações. É também possível, renovar ou eliminar as subscrições. A operação *Subscribe* tem um filtro idêntico ao do *Enumerate*, em que se usam predicados booleanos ou outras linguagens como SQL ou CQL para definir os eventos a subscrever. Em resposta a uma subscrição é enviado para o subscritor o *Endpoint Reference* de um gestor de subscrições que pode ser um *web service* distinto do gerador de eventos. Um problema típico da subscrição de eventos ocorre quando se passa muito tempo sem que ocorram eventos, pelo que o subscritor não sabe se não estão a ocorrer eventos ou se existe alguma anomalia na subscrição ou na ligação que impede a receção dos mesmos. Para tal, o WS-MAN define *heartbeats,* que são pseudo eventos enviados periodicamente para informar que a subscrição ainda está ativa e a funcionar. Para garantir a receção de todos os eventos o subscritor pode também subscrever *bookmarks* de um evento. Esta funcionalidade necessita que existam registos

(*logs*) atualizados dos eventos por parte do gerador de eventos. Estes *bookmarks* são enviados sempre que ocorre um evento e indicam a posição no *log*. Com esta informação o subscritor consegue perceber se houve falha da receção de algum evento. O WS-MAN define também a operação *GetStatus* que no caso de a subscrição estar ativa, implica que o gestor de subscrições envie uma resposta com a data e o tempo de validade da mesma. Para terminar uma subscrição existe a operação *Unsubscribe* que elimina a subscrição. Em caso de falha na entidade que envia notificações deve ser enviada a mensagem *SubscriptionEnd*, para informar o subscritor das notificações. De forma semelhante ao *Enumeration*, está disponível a operação *Renew* que renova o prazo de validade da subscrição. O gestor de subscrições pode recusar uma subscrição quando não são respeitados os requisitos de segurança, como a autorização de acesso a determinado recurso. Os *delivery modes* são outra característica essencial do WS-MAN, permitem definir diferentes mecanismos para transmissão de notificações. Por exemplo, pode-se definir a transmissão síncrona (PushWithAck), assíncrona (Push) ou de conjuntos de notificações (Batched) numa só mensagem para reduzir o tráfego na rede.

Uma vez que o acesso indevido aos recursos de uma rede podem ter efeitos catastróficos para a prestação dos serviços, o WS-MAN permite negociar mecanismos de autenticação e credenciais. Por exemplo, é possível usar HTTPS em detrimento do HTTP ou indicar o uso de certificados para autenticação. A norma também prevê a implementação de novas operações com parâmetros de entrada e de saída, bem definidos, através da linguagem WSDL.

O WS-MAN pelas suas características e funcionalidades conta com implementações em diversas plataformas e SOs. O *Integrated DELL Remote Access Controller* (iDRAC6) [97] é um controlador da DELL para gestão de hardware e software de um sistema, incluindo uma implementação do WS-MAN para gestão remota [98]. O iDRAC6 mapeia *profiles* do DMTF e alguns implementados pela própria DELL. A *Microsoft* disponibiliza nas versões *Server* e *Vista* do seu SO o *Windows Remote Management* (WinRM) [99] e o *Windows Remote Shell* (WinRS). O *WinRM* é a implementação da *Microsoft* do WS-MAN, que através de scripts executa operações do protocolo para adquirir dados de gestão. Esta aplicação obtém os dados do WMI, já mencionado no capítulo anterior. O WinRS é a aplicação de gestão cliente. A Intel tem uma solução de hardware para gestão, o *Active Management Technology* [100], que inclui *WS-MAN Translator* [101] para permitir a utilização de CIM schemas e extensões implementadas pela própria Intel nas suas operações, permitindo também a interação com outras aplicações WS-MAN, como o WinRM [102]. Também a Novell em 2007, numa iniciativa de colaboração técnica com a Microsoft [103], desenvolveu em cooperação com a comunidade suporte open-source para WS-MAN no *Novell ZENWorks Orchestrator*. Actualmente o *openSUSE* e o *SUSE Linux Enterprise Server* incluem suporte para WS-MAN. O OpenWSMAN [104] é a implementação *open-source* do WS-MAN que surgiu da cooperação com a *Novell*. Este projeto resultou na implementação completa de um servidor e um cliente para gestão de plataformas Unix/Linux, totalmente compatíveis com a norma do DMTF. Um outro projeto *open-source* é o *A Java Implementation of WS-Management* [105], focado em

interoperabilidade, devido às características da linguagem de programação Java. Também o projeto OpenPegasus implementou suporte para WS-MAN [106] na sua implementação WBEM.

Embora o MUWS e o WS-MAN tenham sido concebidos para gestão de sistemas, são perfeitamente aplicáveis à gestão de redes. Inicialmente temia-se que a verbosidade do XML seria impeditiva da sua aplicação para gestão. No entanto, alguns estudos mostram que os dados de gestão representados em XML podem ser eficientemente comprimidos [74] [107] [108]. A utilização de compressão e de mecanismos de segurança têm consequências como o aumento dos tempos de resposta nas trocas de mensagens e o aumento dos requisitos computacionais, como o CPU.

O MUWS é tido como uma implementação mais robusta mas também mais pesada, enquanto o WS-MAN apresenta uma norma mais leve e usa um conjunto mais reduzido de operações, consequentemente as suas implementações apresentam melhor desempenho [109].

2.3. NETCONF - Network Configuration Protocol

O NETCONF é o resultado do esforço do IETF em criar um protocolo para configuração e monitorização de redes com dispositivos de diferentes fabricantes. O protocolo permite criar, manipular e apagar configurações dos dispositivos e fornece suporte para operações de monitorização. No desenvolvimento do NETCONF pretendeu-se criar uma solução que permita distinguir dados de configuração e de estado, que seja suficientemente extensível e programável para que os fornecedores de equipamento consigam dar acesso aos dados de configuração através de um único protocolo, que use uma representação de dados baseada em XML, que permita a integração de mecanismos de segurança e bases de dados, que suporte vários repositórios de dados, que integre mecanismos para transações das configurações, que suporte diferentes protocolos seguros de transporte e que suporte notificações assíncronas de eventos. O seu desenvolvimento foi independente da linguagem de modelação de dados, mas o IETF recomenda o YANG, que é uma linguagem desenvolvida especificamente para a gestão de configurações.

Assim, o NETCONF segue uma arquitetura cliente-servidor, estando conceptualmente dividido em quatro camadas apresentadas na **Figura 15**. O cliente solicita operações sobre as configurações alojadas no servidor e pode subscrever notificações de eventos do mesmo. O servidor executa as operações requeridas pelos clientes e envia notificações de eventos quando houver subscrições.

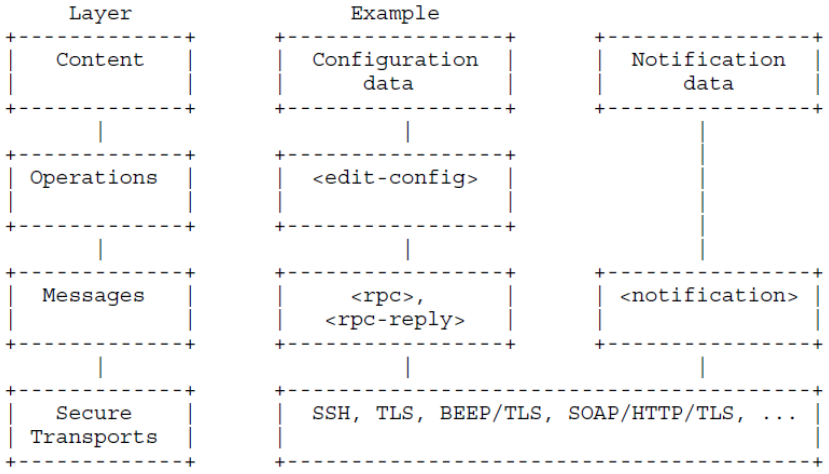

Figura 15. Camadas do NETCONF (retirada da RFC 4741bis-10)

É utilizada a linguagem YANG para descrever tanto as configurações dos dispositivos de rede como as mensagens do protocolo. Para a comunicação entre cliente e servidor usam-se mensagens *Remote Procedure Call*s (RPC) encapsuladas num protocolo de transporte. Pretendia-se, que o protocolo de transporte fosse do tipo "*connection-oriented*", para garantir a persistência da ligação, para providenciar a entrega ordenada de pacotes e garantir a fiabilidade das comunicações. A camada de transporte é responsável pela autenticação, integridade e confidencialidade da ligação entre cliente e o servidor, para isso estão previstas quatro opções para o transporte de informação: o SSH[22], o BEEP[23], o SOAP[24] e o TLS[110]. A RFC 4741 impõe a utilização do SSH, pelo que dois *peers* de NETCONF que pretendam comunicar fazem-no primeiramente em SSH e só depois, caso pretendam, explicitam a intenção de utilizar outro protocolo no processo de troca de capacidades durante a negociação da sessão de NETCONF. Nessa negociação, o cliente envia as capacidades encapsuladas numa mensagem <*hello*> e o servidor responde também com uma mensagem <*hello*> com as capacidades suportadas, comuns às do cliente.

A comunicação entre cliente e servidor faz-se através de mensagens <*rpc*> e <*rpc-reply*>. O cliente faz pedidos ao servidor sob a forma de mensagens <*rpc*>, e os <*rpcs-reply*> são enviados pelo servidor em resposta aos pedidos do cliente. Ambas as mensagens têm um <*message-id*> obrigatório que permite associar pedidos e respostas. Em caso de sucesso, o servidor inclui no <*rpc-reply*> um elemento <*ok*> indicando o sucesso do pedido; quando ocorrem erros o servidor inclui na resposta um ou mais elementos <*rpc-error*> para sinalizar o erro ocorrido. Um <*rpc-error*> tem parâmetros definidos pelo protocolo como <*error-type*>, <*error-tag*>, <*error-severity*>, <*error-message*> e <*error-info*>. As principais fases e trocas de mensagens do protocolo NETCONF estão ilustradas na **Figura 16**.

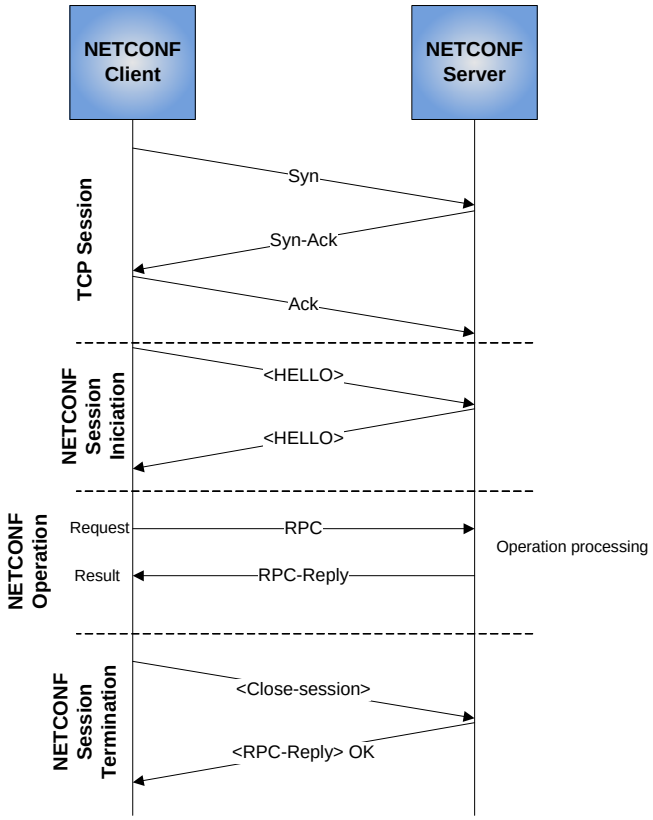

Figura 16. Fases e trocas de mensagens do protocolo NETCONF

Na camada de operações são implementadas as operações definidas no NETCONF, que podem ser solicitadas pelo cliente ao servidor. Estas operações permitem pedir, manipular, copiar e apagar configurações, bem como visualizar informação de estado de um dispositivo. É ainda possível definir operações adicionais através de capacidades extra anunciadas pelos clientes e servidores.

Assim as operações do NETCONF definidas na RFC 4741 e na RFC 5277 estão representadas na **Tabela 4.** Em complemento, a **Tabela 5** apresenta os parâmetros que devem ser usados em cada operação.

Tabela 4. Operações do NETCONF

Operações	***Namespace***	***Descrição***
get-config	:base	Devolve a *"running config"* e a informação de estado do servidor.
edit-config	:base	Permite editar ou criar, caso não exista, uma configuração. Esta alteração pode ser definida nó a nó, mediante a especificação de um atributo *"operation"* diferente em cada nó, a operação por omissão é sempre a *"merge"*
copy-config	:base	Copia uma configuração inteira para outra. Caso o objeto de destino não exista, ele é criado
delete-config	:base	Apaga uma configuração. A configuração *"running"* não pode ser apagada
lock	:base	Permite que um cliente bloqueie o acesso a uma configuração. Durante este bloqueio a configuração pode ser livremente alterada sem

		influenciar ou ser influenciada pela interação de outros clientes. O bloqueio termina quando explicitado pelo cliente ou terminar a sessão com o mesmo
unlock	:base	Desbloqueia o acesso a uma configuração
get	:base	Esta operação devolve a *"running config"* e a informação de estado do servidor
close-session	:base	Pede para terminar a sessão com um servidor. O servidor liberta todos os *locks* e recursos alocados nesta sessão e termina a ligação
kill-session	:base	É forçado o fim de uma sessão NETCONF com o servidor. As operações desta sessão são abortadas e todos os recursos são desbloqueados
commit	:candidate	Aplica a configuração *candidate* à configuração *running*
discard-changes	:candidate	Copia a configuração *running* para a configuração *candidate* sem aplicar alterações à configuração *running*
cancel-commit	:confirmed-commit	Cancela um *commit* em execução
validate	:validate	Verifica o conteúdo de uma configuração
create-subscription	:notification	Esta operação permite registar a intenção de um cliente em receber notificações assíncronas de eventos. Existe a possibilidade de pedir notificações antigas através da funcionalidade de *replay*. Esta operação é essencial ao processo de monitorização do NETCONF, descrito na **Figura 27**.

As operações descritas na **Tabela 4** usam combinações dos parâmetros que se seguem:

- *source*: Especifica a configuração de origem dos dados, Ex.: *candidate;*
- *target*: Especifica a configuração a editar, Ex.: *startup;*
- *filter*: Um dos parâmetros mais importantes do NETCONF, permite indicar um filtro para selecionar apenas parte dos dados de configuração ou de estado. Caso este campo esteja vazio, todos os dados são devolvidos. O filtro tem um atributo *"type"* que permite indicar o tipo de pesquisa a fazer. Em caso de omissão usa-se a pesquisa *"subtree"*, caso exista suporte da capacidade *"XPath"* pode-se definir este campo como *"xpath"* para efectuar uma filtragem deste tipo;
- *default-operation*: Define o tipo de manipulação a efetuar pela operação *edit-config* sobre uma configuração, podendo tomar um de três valores:
  - ✓ *merge*: as configurações do campo *config* são adicionadas na configuração *target;*
  - ✓ *replace*: as configurações em *config* sobrepõem-se às da configuração *target*;
  - ✓ *none:* as configurações em *config* são ignoradas até que seja utilizado o atributo *operation* a indicar uma operação diferente;
- *test-option*: Este parâmetro pode ser especificado se for anunciada a capacidade *:validate* e toma um dos seguintes valores*:*

  - ✓ *test-then-set:* Inicialmente é feita a validação de *config*, caso não se verifiquem erros as alterações são aplicadas;
  - ✓ *set:* As alterações são aplicadas sem validação das configurações;
  - ✓ *test-only:* Apenas é feita a validação de *config* sem aplicar as configurações;
- *error-option*: a política de tratamento de erros, com os seguintes valores possíveis:
  - ✓ *stop-on-error:* Assim que houver um erro a edição das configurações é abortada. Esta é a opção por omissão;
  - ✓ *continue-on-error:* A aplicação das alterações continua mesmo em caso de erro. É gerada uma resposta conforme os erros ocorridos;
  - ✓ *rollback-on-error:* Em caso de erro as alterações são abortadas e as configurações são restauradas até ao estado do inicio da alteração. Esta opção é disponibilizada pela capacidade *":rollback-on-error"*;
- *config*: Especifica as alterações a fazer, tem que estar de acordo com o modelo de dados e *namespace* do *target;*
- *session-id*: Identificador da sessão a terminar;
- *stream*: Identificador de um fluxo de eventos. São conjuntos de notificações que respeitam um dado critério. Na omissão deste parâmetro, são enviadas notificações da *stream* por omissão que normalmente inclui todas as notificações de um agente;
- *start time:* subscreve notificações a partir de um instante temporal. Faz parte da funcionalidade de *replay* da capacidade :notification;
- *stop time*: também faz parte da funcionalidade de *replay* e deve ser utilizado em conjugação com o parâmetro anterior para definir o intervalo de tempo de que se quer receber notificações;
- *persist-id:* valor identificador de um *commit*, só é válido mediante a capacidade *:confirmed-commit*.

Em caso de erro nas operações o servidor envia um rpc-error (**Caixa de Código 5**) com uma mensagem de erro.

Por fim, sobre a camada de operações estão representados os modelos de dados do dispositivo que contém o servidor. Para representar esses modelos é usada a linguagem YANG[111], desenvolvida pelo *NETMOD Work Group* [112] para descrição dos dados e operações de gestão do NETCONF. É uma linguagem de modelação de dados capaz de representar os dados de configuração, os dados de estado, as mensagens RPC e as notificações do protocolo. Recorrendo ao YANG, definem-se os dados, a sua organização hierárquica e as suas restrições. O YANG define ainda uma representação XML dos dados e a sua utilização nas operações do NETCONF através do seu equivalente XML, designada por YANG Independent Notation (Yin). Os módulos YANG funcionam como contratos entre cliente e servidor; após acordarem quais os módulos a usar, ambos sabem como proceder e o que esperar por parte do *peer*.

Tabela 5. Parâmetros das operações do NETCONF

Operação	**SessionID**	**Filter**	**Config**	**Target**	**Source**	**Default-operation/ Error Opt**	**Resposta quando sucesso**
get-config	*Não*	*Obr*	*Não*	*Não*	*Obr*	*Não*	*<data>*
edit-config	*Não*	*Opc*	*Obr*	*Obr*	*Não*	*Opc/Obr*	*<ok>*
copy-config	*Não*	*Não*	*Não*	*Obr*	*Obr*	*Não*	*<ok>*
delete-config	*Não*	*Não*	*Não*	*Obr*	*Não*	*Não*	*<ok>*
lock	*Não*	*Não*	*Não*	*Obr*	*Não*	*Não*	*<ok>*
unlock	*Não*	*Não*	*Não*	*Obr*	*Não*	*Não*	*<ok>*
get	*Não*	*Opc*	*Não*	*Não*	*Não*	*Não*	*<data>*
close-session	*Não*	*Não*	*Não*	*Não*	*Não*	*Não*	*<ok>*
kill-session	*Obr*	*Não*	*Não*	*Não*	*Não*	*Não*	*<ok>*
Validate	*Não*	*Não*	*Não*	*Não*	*Obr*	*Não*	*<ok>*
	Persist-id	**Filter**	**Stream**	**Start time**	**Stop time**		
commit	*Não*	*Não*	*Não*	*Não*	*Não*		*<ok>*
discard-changes	*Não*	*Não*	*Não*	*Não*	*Não*		*<ok>*
cancel-commit	*Opc (SessionID)*	*Não*	*Não*	*Não*	*Não*		*<ok>*
Validate	*Não*	*Não*	*Não*	*Não*	*Não*		*<ok>*
create-subscription	*Não*	*Opc*	*Opc*	*Opc*	*Opc*		*<ok>*

O YANG valoriza a fácil e rápida legibilidade por humanos e, quando comparado com XML schema ou SMIng, demonstra claras vantagens para ser usado com o NETCONF [113]. Os modelos de dados do YANG apresentam uma organização hierárquica em árvore e uma estrutura modular que permite importação e exportação de informação entre módulos, tornando-a numa linguagem extremamente flexível e extensível, facilitando a integração de modelos de dados normalizados ou proprietários. Podem-se usar módulos para adaptar as funcionalidades existentes noutros (*augment*), ou também é possível definir novas funcionalidades.

A representação hierárquica do XML permite a representação de dados e das suas interdependências e relações. A **Figura 17**, mostra a atribuição de interfaces a áreas OSPF. O elemento *<ospf>* tem uma lista de áreas *<area>*, que por sua vez contém uma lista das interfaces *<interface>* que constituem essa área. O elemento <name> serve de identificador de cada um destes elementos e dentro da definição de cada interface estão configurações específicas que só a ela dizem respeito [114].

```
<ospf xmlns="http://example.org/netconf/ospf">

  <area>
    <name>0.0.0.0</name>

    <interface>
      <name>ge-0/0/0.0</name>
      <!-- The priority for this interface -->
      <priority>30</priority>
      <metric>100</metric>
      <dead-interval>120</dead-interval>
    </interface>

    <interface>
    <name>ge-0/0/1.0</name>
    <metric>140</metric>
    </interface>
  </area>

  <area>
    <name>10.1.2.0</name>

    <interface>
      <name>ge-0/0/2.0</name>
      <metric>100</metric>
    </interface>

    <interface>
      <name>ge-0/0/3.0</name>
      <metric>140</metric>
      <dead-interval>120</dead-interval>
    </interface>
  </area>
</ospf>
```

Figura 17. Exemplo em YANG da atribuição de interfaces a diferentes áreas OSPF

O YANG oferece um vasto conjunto de tipos de dados pré definidos [115], que normalmente não se encontram noutras linguagens, como por exemplo "ipv4-address" (módulo "ietf-inet-types"), e em opção é também possível definir novos tipos de dados. Para construir os modelos de dados o YANG apresenta um conjunto de diretivas, descritas na **Tabela 6**, que permitem a representação hierárquica dos dados e a sua reutilização.

Tabela 6. Principais directivas do YANG

Directiva	*Descrição*
Augment	Serve para estender uma hierarquia de dados existente
Choice	Define nós mutuamente exclusivos
Container	Permite agregar um conjunto de nós numa sub-árvore
Extension	Permite adicionar novas diretivas ao modelo de dados
Feature	Permite indicar partes opcionais do modelo de dados
Grouping	Permite agrupar as definições de dados em conjuntos reutilizáveis
Key	Define o identificador das entradas de uma lista
Leaf	Define um nó folha na hierarquia de dados
Leaf-list	Define uma lista de nós folha
List	Permite definir uma lista de nós
Notification	Permite definir os dados de uma notificação
Rpc	Define os parâmetros de entrada e saída de uma operação RPC
Type	Restringe o conteúdo de um nó folha a um tipo de dados
Typedef	Define um novo tipo de dados
Uses	Permite usar os conjuntos definidos com a diretiva *grouping*

Assim, recorrendo a estas diretivas é possível construir módulos YANG como o exemplificado na **Figura 18**.

```
// Contents of "acme-system.yang"
module acme-system {
    namespace "http://acme.example.com/system";
    prefix "acme";

    organization "ACME Inc.";
    contact "joe@acme.example.com";
    description
        "The module for entities implementing the ACME system.";

    revision 2007-06-09 {
        description "Initial revision.";
    }

    container system {
        leaf host-name {
            type string;
            description "Hostname for this system";
        }

        leaf-list domain-search {
            type string;
            description "List of domain names to search";
        }

        container login {
            leaf message {
                type string;
                description
                    "Message given at start of login session";
            }

            list user {
                key "name";
                leaf name {
                    type string;
                }
                leaf full-name {
                    type string;
                }
                leaf class {
                    type string;
                }
            }
        }
    }
}
```

Figura 18. Exemplo de um módulo YANG

Também é possível definir restrições aos dados para prevenir dados impossíveis ou que não façam sentido. Estas restrições podem ser aplicadas aos dados de configuração das operações RPC e das notificações. A principal restrição é aplicada com a diretiva *type*, mas o YANG define outras enumeradas na **Tabela 7**.

Tabela 7. Restrições implementadas pelo YANG

Restrição	*Descrição*
Lenght	Limita o tamanho de uma string
Mandatory	Torna um nó obrigatório
Max-elements	Limita o tamanho máximo de entradas numa lista
Min-elements	Limita o tamanho mínimo de entradas numa lista
Must	Impõe que uma expressão XPath seja verdadeira
Pattern	Aplica uma expressão regular
Range	Define um intervalo de valores aceitável
Reference	Referência a um valor
Unique	Torna o valor único nos dados
When	Um nó só exite quando é satisfeita uma expressão XPath

Para satisfazer as necessidades do NETCONF, o YANG disponibiliza algumas funcionalidades como distinguir dados de estado e configuração, através da diretiva *config* (**Figura 19**), ou prevê a possibilidade do dispositivo indicar que não implementa completamente um modelo de dados, através da diretiva *deviation*. A declaração desta diretiva permite identificar limitações e assim, evitar erros em *run-time*.

```
leaf observed-speed {
    type uint32;
    config false;
}
```

Figura 19. Distinção entre dados de configuração e de estado em YANG

Para ambientes com recursos limitados onde não é viável ter um parser de YANG, está prevista uma linguagem XML, denominada *YANG Independent Notation* (Yin). O Yin permite usar *parsers* de XML, que facilmente se implementam para operar os modelos de dados. As conversões entre YANG e Yin (**Figura 20**) preservam a estrutura e conteúdo YANG mantendo uma equivalência semântica. Está também prevista a conversão para *Document Schema Definition Language* (DSDL), para facilitar a validação XML, utilizando esta linguagem amplamente difundida.

A linguagem YANG oferece uma flexibilidade e extensibilidade ímpar para a descrição de dados e operações para gestão. É por isso, a mais adequada para representar o modelo de dados do NETCONF.

```
YANG module:

module acme-foo {
   namespace "http://acme.example.com/foo";
   prefix "acfoo";

   import my-extensions {
       prefix "myext";
   }

   list interface {
       key "name";
       leaf name {
           type string;
       }

       leaf mtu {
           type uint32;
           description "The MTU of the interface.";
           myext:c-define "MY_MTU";
       }
   }
}
```

```
Yin Tanslation:

<module name="acme-foo"
        xmlns="urn:ietf:params:xml:ns:yang:yin:1"
        xmlns:acfoo="http://acme.example.com/foo"
        xmlns:myext="http://example.com/my-extensions">

 <namespace uri="http://acme.example.com/foo"/>
 <prefix value="acfoo"/>

 <import module="my-extensions">
   <prefix value="myext"/>
 </import>

 <list name="interface">
   <key value="name"/>
   <leaf name="name">
     <type name="string"/>
   </leaf>
   <leaf name="mtu">
     <type name="uint32"/>
     <description>
       <text>The MTU of the interface.</text>
     </description>
     <myext:c-define name="MY_MTU"/>
   </leaf>
 </list>
</module>
```

Figura 20. Conversão YANG para Yin

O protocolo NETCONF define a existência de um ou mais repositórios de configurações. Os repositórios são conjuntos de configurações que permitem a um dispositivo mudar do estado inicial para um estado operacional. Estão definidos três tipos de configurações: uma configuração *running* que é a configuração em vigor durante o funcionamento do dispositivo, uma configuração *startup* que tem as configurações por omissão carregadas no arranque do dispositivo e uma configuração *candidate*, que permite manipular uma configuração sem alterar o estado atual do equipamento.

O NETCONF permite extensões às funcionalidades previstas no protocolo através de capacidades que podem, por exemplo, ser definidas por fabricantes de equipamento de rede para adaptar o protocolo às funcionalidades específicas dos diferentes equipamentos. Estas capacidades são anunciadas pelo cliente e pelo servidor aquando do estabelecimento da sessão; nessa altura são negociadas as capacidades a usar durante a sessão. As

configurações *startup* e *candidate* não fazem parte do modelo de dados base e podem por isso, ser anunciadas no processo de anúncio de capacidades.
A RFC 4741 propõe um conjunto de capacidades para o protocolo:

- *:writable-running* – Permite a alteração da configuração *running* em *runtime* **(Figura 21)**;
- *:candidate* – Indica que o dispositivo suporta a configuração *candidate* e duas novas operações <*commit*> e <*discard-changes*> **(Figura 21)**;
- *:confirmed-commit* – Esta capacidade é dependente da :*candidate* e oferece um conjunto de funcionalidades para garantir a consistência dos *commits;*
- *:rollback-on-error* – torna possível esta opção no parâmetro <*error-option*> da operação <*edit-config*>;
- *:validate* – torna possível validar erros semânticos e de sintaxe tanto, através de uma opção do parâmetro <*test-option*> da operação <*edit-config*> ou através da operação <*validate*>;
- *:startup* – Indica o suporte da configuração *startup;*
- *:url* – Permite usar *Uniform Resource Locators* (URL) para definir <sources> e <targets> das operações, bem como a opção <*config*> da operação <*edit-config*>;
- *:xpath* – permite usar expressões XPath no filtro de pesquisa;
- :with-defaults – permite definir valores por omissão entre um cliente e um servidor. Neste caso o servidor não precisa de guardar estes dados nas configurações nem nas respostas ao cliente [116].

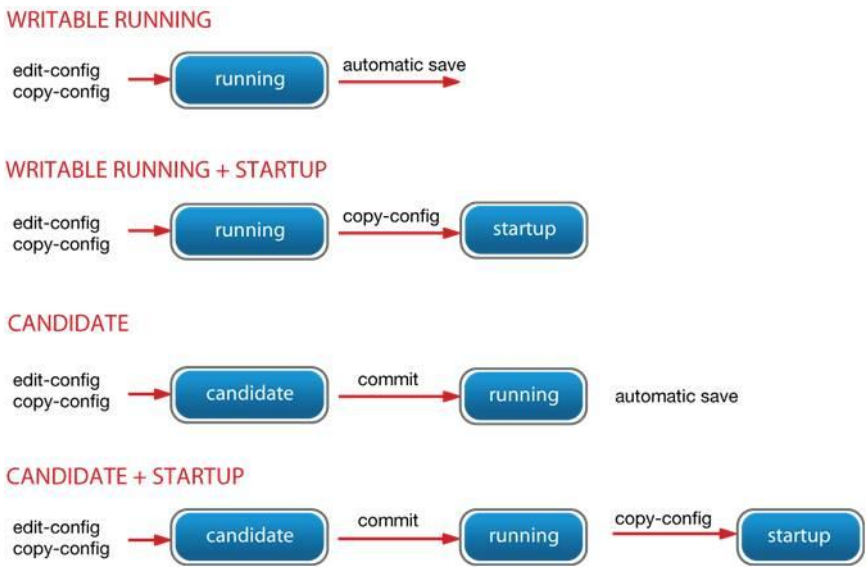

Figura 21. Transações permitidas pelas capacidades *writable-running* e *candidate* [117]

O XML é usado pelo NETCONF para descrever as configurações dos dispositivos e as mensagens RPC trocadas pelos clientes e pelo servidor. Todas as mensagens do NETCONF têm que respeitar a norma XML e serem codificadas em UTF-8, caso contrário devem ser rejeitadas pelo servidor.

Os elementos e atributos do NETCONF estão definidos no "*namespace*":

"urn:ietf:params:xml:ns:netconf:base:1.0"

Também as capacidades do NETCONF têm de ser definidas por URIs, como por exemplo:

"urn:ietf:params:netconf:capability:startup:1.0"

O NETCONF é um protocolo recente, que tem vindo a aumentar a sua aceitação por parte dos administradores de redes. Por isso, conta com várias implementações *open-source* e algumas proprietárias, bem como suporte em aplicações de algumas das maiores marcas do mercado, como a *Cisco* ou a *Juniper*.

Uma das implementações *open-source* mais divulgadas é o *Yuma* [118]. O *Yuma* é um pacote de ferramentas que inclui uma aplicação cliente e uma aplicação servidor de NETCONF sobre SSH e ferramentas para manipular módulos YANG, como um compilador e um comparador de módulos. Esta implementação encontra-se muito bem documentada e suporta um conjunto muito diversificado de capacidades NETCONF e módulos YANG. Existe também uma ferramenta para *Android*, o *netconf4android* [119]. É uma API em Java para gerar aplicações cliente de NETCONF. Funciona sobre SSH mas permite ao utilizador integrar outras tecnologias de transporte. E integra suporte para mensagens assíncronas para monitorização.

O *netopeer*[120] é uma implementação *open-source* do NETCONF desenvolvido pelo CESNET ao abrigo do projecto *FlowMon Probe* [121] e posteriormente do *Flexible FlowMon Probe* [122], ambos desenvolvem circuitos para monitorização de redes. Assim, o *netopeer* foi desenvolvido para, usando o NETCONF, enviar instruções e receber os relatórios/notificações da operação destes dispositivos. Já se encontra em desenvolvimento uma nova versão compatível com as RFCs lançadas durante o ano de 2011.

O Yenca[123], terá sido a primeira implementação do NETCONF. Lançado em 2004, consistia num agente desenvolvido em C e um gestor em Java. Foi desenvolvido pelos laboratórios LORIA-INRIA que desenvolveram a sua arquitetura dando origem ao EnSuite [124]. O EnSuite é um protótipo de uma *framework* de NETCONF que fornece uma plataforma *open-source* para testes, não só do NETCONF como também do YANG. Foi desenvolvido em python e inclui uma aplicação de gestão web-based, o YencaP-Manager, e um agente com suporte para adição de novos módulos e operações, o YencaP.

Também desenvolvido em python existe o *ncclient* [125]. É uma biblioteca que mapeia a natureza XML do NETCONF para python, a fim de facilitar a escrita de *scripts* para a gestão de redes. Suporta todas as operações e capacidades da RFC 4741. O *ncclient* tem uma boa compatibilidade com outras implementações como o netopeer, o *ConfD* da Tail-f e o IOS 12.4 da Cisco.

No domínio das aplicações proprietárias, a empresa *Tail-f Systems* [126] apresenta várias soluções que tiram partido do NETCONF. A Tail-f oferece uma aplicação cliente (*ConfM*) e outra servidor (*ConfD*), com suporte não só para NETCONF mas também para SNMP, CLI e Web UI. Oferece também, uma framework (NCS) para desenvolvimento de sistemas de gestão usando NETCONF. De notar, que a Tail-f oferece formação em NETCONF e YANG o que demonstra o "entusiasmo" em relação a estas tecnologias.

A empresa *GoAhead* apresenta um conjunto de ferramentas, o *embeddedMIND* [127], que permite aos fabricantes incluir agentes de gestão nos seus equipamentos de rede. Este pacote é constituído por uma framework, a *MINDframework*, que é uma infra-estrutura de gestão que suporta *MINDConfStore*, *MINDObjects* e *MINDAgents*. O *MINDConfStorage* é uma solução para preservar configurações, *logs* e dados de acesso, de forma independente do protocolo de transporte. O MINDObjects é responsável pela representação e conversão dos dados e, finalmente, os MINDAgents são APIs de agentes de NETCONF, SNMP, CLI e Web UI.

Alguns dos maiores fabricantes de equipamento de rede já integram o NETCONF nas suas soluções. A Cisco integra no seu IOS um agente de NETCONF e suporta transporte sobre SSHv2 [128] e sobre BEEP [129], bem como uma interface gráfica de gestão, o *NETCONF Client GUI* [130].

A *Juniper* inclui um servidor de NETCONF no Junos OS e oferece o NETCONF XML Management Protocol (uma adaptação do NETCONF para o Junos OS CLI), um modulo em perl para aplicações cliente e uma API para desenvolvimento personalizado de clientes de gestão [131].

2.3.1. Filtro "Subtree"

O filtro "*subtree*" é uma funcionalidade importante do NETCONF, que permite procurar uma sub-árvore dentro do modelo de dados XML do servidor.

Esta funcionalidade é especialmente útil nas repostas a <*get*> e <*get-config*>, onde tipicamente se quer visualizar parte de uma configuração. O filtro percorre os nós duma configuração comparando elementos e atributos, devolvendo apenas a sub-árvore cujos elementos respeitem o filtro de pesquisa.

Um filtro *"subtree"* pode pesquisar cinco tipos de elementos distintos ou uma combinação destes:

- *"Namespace":* Esta pesquisa compara nós ou ramos definidos dentro de um mesmo *"namespace"*. É de realçar que não existem pesquisas de *"namespace"* por si só, é sempre necessário definir um nó para fazer a correspondência dentro do *"namespace"*. Esta pesquisa pode ser considerada um caso específico de uma pesquisa por nó;
- Atributos: É feita uma pesquisa por nós que tenham um dado atributo. Para existir correspondência é necessário que a árvore que dá acesso ao nó seja igual;
- Ramos: Neste tipo de pesquisa, procura-se por nós que contêm nós filhos, quando existe uma correspondência, na resposta consta o nó e a sua subárvore;

- Nós terminais*:* Esta pesquisa é equivalente à anterior excetuando que se procuram nós terminais e, como tal, em caso de sucesso só são devolvidos os atributos do nó;
- Atributos dentro de Nós terminais*:* Esta pesquisa procura todos os nós que tenham um atributo específico.

Nas pesquisas "subtree" é necessário especificar todos os nós desde a raiz, obrigando o cliente a conhecer na totalidade o modelo de dados do servidor. Isto implica perda de funcionalidade e ganho no desempenho quando comparado com outros métodos de pesquisa explicados mais à frente. No entanto, este tipo de pesquisa é tido como o mais simples para o utilizador.

Em alternativa a pesquisas usando um filtro *"subtree"*, o NETCONF permite pesquisas *"XPath"* usando **XPath 1.0,** como uma capacidade com o identificador,

"urn:ietf:params:netconf:capability:xpath:1.0"

O *"XPath"* é uma norma do W3C [93], que utiliza expressões semelhantes aos caminhos de um sistema de ficheiros para navegar num objeto XML. Permite procurar um ou mais nós comparando o caminho com a árvore de nós de um objeto XML e utiliza predicados para identificar nós com atributos específicos. As respostas das pesquisas com *"XPath"* são sub-árvores XML que respeitam o filtro de pesquisa.

A pesquisa *"XPath"* não necessita do caminho absoluto para efetuar uma pesquisa, não sendo necessário conhecer todo o modelo de dados do servidor, tornando-se mais flexível e funcional. Contudo, as expressões *"XPath"* são mais complexas e difíceis de aprender, ficando em desvantagem em relação a pesquisas *"subtree"* que são mais simples para o utilizador.

O método de pesquisa *"XPath"* apresenta um desempenho muito inferior ao do método *"subtree"* [132]. Isto deve-se ao facto de o *"subtree"* só ter que comparar o caminho indicado no filtro de pesquisa, enquanto, no *"XPath"* é necessário comparar todo o modelo de dados, o que leva a uma rápida perda de eficiência à medida que o modelo de dados aumenta.

2.4. Sumário

Neste capítulo descreveram-se várias tecnologias de gestão e monitorização de redes. O SNMP foi descrito por ser um dos protocolos mais antigos para SGR e sem dúvida o mais utilizado até ao momento. Entretanto, com a evolução das redes de comunicação, o SNMP já não é suficiente para uma gestão e monitorização eficaz de redes cada vez maiores e mais complexas. Assim, têm-se vindo a adotar tecnologias como o HTTP e o XML para resolver os problemas da gestão de redes. Atendendo a este facto, as

tecnologias *web based* e sobre *web services* têm vindo a ganhar cada vez mais aceitação e começam a demarcar-se como tecnologias ímpares para a gestão e monitorização de redes. Por isso, para este trabalho foi relevante entender abordagens como o WBEM, o WSDM e o WS-MAN. Por fim surge o NETCONF, um protocolo novo e inovador com uma visão mais flexível e personalizável para facilitar a gestão e monitorização de equipamentos de rede.

De seguida apresentam-se duas tabelas (**Tabela 8** e **Tabela 9**) comparativas destas tecnologias, de forma a simplificar a sua análise.

Tabela 8. Comparação de Linguagens de descrição de dados

Protocolo	*Linguagem*	*Permite definir estruturas de dados*	*Permite definir Operações*	*Permite distinguir dados de estado e de configuração*
SNMP	SMI/ASN.1	N	N	N
WBEM	CIM/XML	S	S	N
WS-MAN	CIM/XML	S	S	N
NETCONF	YANG/Yin/XML	S	S	S

Nas linguagens de modelação de dados é notória a falta de flexibilidade e adaptabilidade do SMI por não permitir definir estruturas de dados ou operações, bem como a identificação de dados de estado ou configuração. A abordagem CIM demonstra ser mais flexível permitindo estas definições mas não permite a distinção entre dados de configuração e estado que agilizam bastante as operações em abordagens orientadas ao documento. A linguagem CIM apresenta um grau de complexidade que dificulta a interoperabilidade tanto ao nível dos *schemas* quando se quer estender as funcionalidades quer entre implementações [11]. O YANG prima pela extensibilidade e adaptabilidade e apresenta duas representações: uma mais indicada para humanos e outra XML mais eficaz para ser usada por máquinas.

Em relação aos protocolos de gestão, opta-se pelos que utilizam protocolos de transporte do tipo orientado à ligação, pela sua fiabilidade e melhores garantias de qualidade nas comunicações. A utilização do XML promove a manipulação de configurações completas e também tira partido do vasto leque de ferramentas de *parsing* existentes o que aumenta a interoperabilidade das implementações. O NETCONF ao implementar as operações de *lock* e mesmo de *partial lock* aumenta bastante a sua operacionalidade em ambientes distribuídos, enquanto implementa mecanismos para transações com funcionalidades como *rollback* que permitem garantir a consistência dos dados. Já a nível de segurança, o SNMP apresenta na sua terceira versão soluções complexas e com fraca integração com soluções já existentes [11], tendo ainda sofrido com a falta de suporte nos dispositivos existentes [42]. Os protocolos de gestão e monitorização sobre HTTP apresentam a possibilidade de utilizar SSL/TLS, que permite a cifra e autenticação dos dados transmitidos. O NETCONF também apresenta uma proposta para controlo de acessos para as operações e conteúdos de gestão [133].

Tabela 9. Comparação dos protocolos descritos na dissertação

Protocolo	*Protocolo de Transporte*	*Gerir configurações completas*	*Notificações*	*Lock/ Partial Lock*	*Extenções*	*Transacções*	*Segurança*
SNMP	UDP	N	S	N/N	N	N	Apenas na v3
WBEM	CIM-XML/ HTTP	S	S	N/N	S	N	HTTPS
WS-MAN	SOAP/ HTTP	N	S	N/N	S	N	HTTPS
NETCONF	SSH, BEEP, SOAP/HTTP, TLS	S	S	S/S	S	S	HTTPS, Controlo de acesso

No que toca a implementações de sistemas de notificações, realçam-se as *traps* do SNMP, o WS-Eventing usado pelo DMTF, o WS-BaseNotification da OASIS [134] e as notificações do NETCONF [135]. A **Tabela 10** apresenta uma breve comparação destas abordagens. O SNMP mantém os problemas relacionados com o uso de UDP no transporte e as poucas classes de eventos que limitam bastante a sua utilização. O WS-Eventing é a norma do WS-* que permite uma filtragem bastante flexível, podendo-se usar várias linguagens baseadas em SQL, com preferência pelo WQL. A funcionalidade de *wrapping* garante a interoperabilidade do sistema podendo enviar notificações para outros sistemas como o SNMP ou o NETCONF e os *bookmarks* permitem sinalizar falhas na receção de notificações. O NETCONF também inclui notificações em que tira partido da filtragem *subtree* e apresenta a possibilidade de fazer subscrições em intervalos de tempo, bem como receber notificações que já ocorreram, permitindo uma boa fiscalização do sistema.

Tabela 10. Comparação de diferentes tecnologias de notificações de eventos.

	SNMP traps	**WS-Eventing (WBEM & WS-MAN)**	**WS-Base Notification**	**NETCONF Notifications**
Âmbito	**Equipamento de rede**	**Genérico**	**Genérico**	**Equipamento de rede**
Framework	**SNMP**	**WS-***	**WSDM**	**NETCONF**
Formato de dados	**ASN.1**	**XML (CIM/XML)**	**XML**	**XML**
Protocolo de transporte	**UDP**	**SOAP (HTTP/HTTPS)**	**SOAP (HTTP/HTTPS)**	**SSH, BEEP, SOAP, TLS**
Suporte para *Web services*	**Não**	**Sim**	**Sim**	**Opcional**
Subscrições	**Não**	**Sim**	**Sim**	**Sim**
Selecção de eventos	**UIDs**	**Filtro(WQL/SQL based) + Wrapper/Handler**	**Filtro (XPath)**	**Fluxos+Filtro (Subtree ou XPath)**
Classes de eventos	**7**	*****	*****	*****
Terminar Automático de Subscrições	**Não**	**Time out**	**Não (Pause/resume)**	**stop time**
Replay de eventos	**Não**	**Só da última notificação (Bookmarks)**	**Só da última notificação**	**Sim**

A utilização de XML é transversal às tecnologias mais recentes de notificação de eventos, facilita o processamento de mensagens, o uso de *web service*s e o desenvolvimento das aplicações de gestão. Também a notificação de eventos associada a *web services* se tornou numa grande vantagem, especialmente em sistemas autonómicos de arquiteturas SOA onde as abordagens tradicionais já não se adequam [134]. Estes sistemas caracterizam-se por serem de grande escala e complexidade, onde é impossível consultar a todo o momento o estado dos seus elementos, por isso a monitorização assíncrona torna-se uma ferramenta incontornável na gestão destes sistemas, com consequências notáveis na sua prestação de serviços.

3. API de monitorização NETCONF

3.1. Requisitos

Existem já algumas implementações do protocolo NETCONF sobre SSH, tendo-se também encontrado uma sobre BEEP mas, até à data de elaboração desta dissertação, não se conseguiu encontrar nenhuma implementação sobre SOAP ou TLS. Os *web service*s têm ganho cada vez mais aceitação no mundo da gestão e monitorização de redes em parte graças à facilidade com que ultrapassam as *firewalls* existentes nas organizações, o que nem sempre acontece quando se utilizam outras alternativas de transporte. Consequentemente, considerou-se pertinente fazer uma implementação do NETCONF sobre SOAP, com as respetivas capacidades de monitorização.

*Web service*s servem para fornecer acesso universal a aplicações de rede usando XML e, segundo a definição do W3C, são sistemas de software identificados por URIs (Universal Resource Identifier) em que as suas interfaces e interações são representadas em XML. As definições destes sistemas podem ser descobertas por outros sistemas. As interações com um *web service* devem ser de acordo com a sua definição e as mensagens devem basear-se em XML e ser transmitidas sobre IP. Baseia-se numa arquitetura em camadas, representada na **Figura 22,** onde normalmente se usa o WSDL para definição dos *web service*s, o SOAP para encapsular as mensagens em XML e HTTP para o transporte. A ubiquidade do HTTP e do XML nos sistemas de informação conferem um alto nível de interoperabilidade no transporte de informação. As críticas a esta abordagem passam pela complexidade e verbosidade do XML e pela necessidade de um servidor HTTP, de um motor de SOAP e de um *parser* de XML. Isto implica o aumento dos requisitos de memória RAM (Random Access Memory), dos requisitos de largura de banda para a comunicação entre elementos de gestão e dos custos de operação [136].

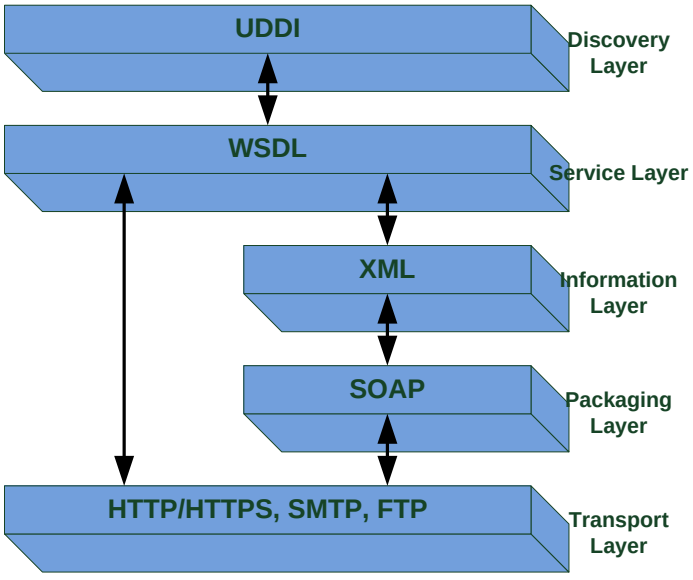

Figura 22. Arquitectura em camadas de um *Web service*

O SOAP é uma norma baseada em XML para troca de informação entre *peers* de forma descentralizada. É uma tecnologia ubíqua, com suporte nos mais variados tipos de dispositivos, desde telemóveis, PDAs, sistemas *embedded* ou aplicações de *desktop*. Tem aplicação em dispositivos com baixos recursos computacionais, como sensores. Pode ser transportado de forma segura usando HTTPS ou utilizar extensões de segurança para, por exemplo, assinaturas digitais ou cifra de informação. Por normalmente usar HTTP é *firewall-friendly*, evitando bloqueios inesperados das comunicações e podendo ser comprimido, contornando a verbosidade apontada como um inconveniente do XML. O SOAP tem ainda suporte para transações e para tratamento remoto de exceções. Uma mensagem SOAP, representada na **Figura 23**, consiste num *envelope* que contém um *header* opcional (para meta informação como dados de autenticação, informação de routing ou tokens de transações) e um *body* com uma representação de um elemento RPC. Os dados transmitidos pelo SOAP podem ser de tipo escalar ou composto. Existe ainda um elemento *SOAP fault* que permite transmitir erros ou exceções.

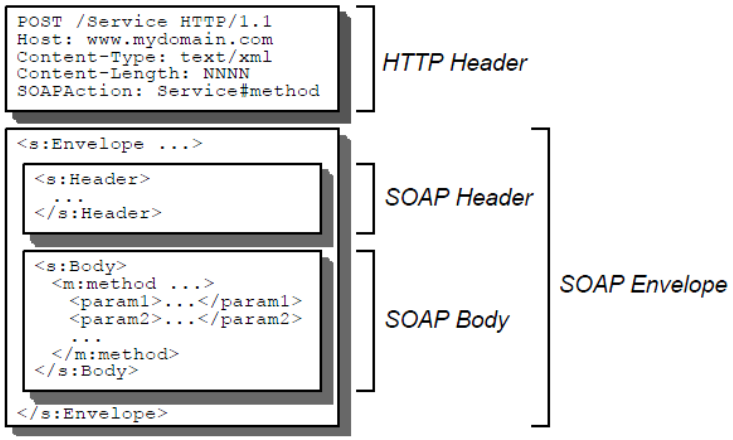

Figura 23. Estrutura de uma mensagem SOAP encapsulada numa mensagem HTTP

Para este projeto optou-se pelas linguagens C/C++, por serem linguagens de baixo nível que proporcionam uma boa gestão de recursos, o que as torna vantajosas para sistemas onde as capacidades operacionais não abundam, como frequentemente é o caso em equipamentos de rede, onde é necessária uma eficiente distribuição de recursos. Isso é visível quando se comparam *frameworks* em C/C++ com *frameworks* em Java, onde se conclui que em média as primeiras demonstram ter melhor desempenho, bem como melhor capacidade de resposta a um maior número de pedidos simultâneos [137].

Consequentemente, e com o SOAP em mente, a framework gSOAP [138] [139] para desenvolvimento de *web service*s foi a escolhida, uma vez que é uma das ferramentas mais completas para o desenvolvimento de *web service*s em SOAP/XML usando C/C++. Também tem suporte para mensagens assíncronas, característica determinante para implementação de notificações. O gSOAP inclui um *parser* de WSDL, o *wsdl2h*, que cria um ficheiro *header*, um compilador RPC, o *soapcpp2*, que gera os stubs/skeletons responsáveis por gerar as mensagens SOAP e uma biblioteca, *stdsoap2*, para serialização e des-serialização em *run-time*. A serialização consiste na conversão de um objeto de dados numa sequência de bits para serem transportados. As mensagens SOAP são geradas através

do processo de *marshaling*. Este processo consiste em codificar os parâmetros duma operação em XML para os encapsular numa mensagem SOAP pronta a transmitir.

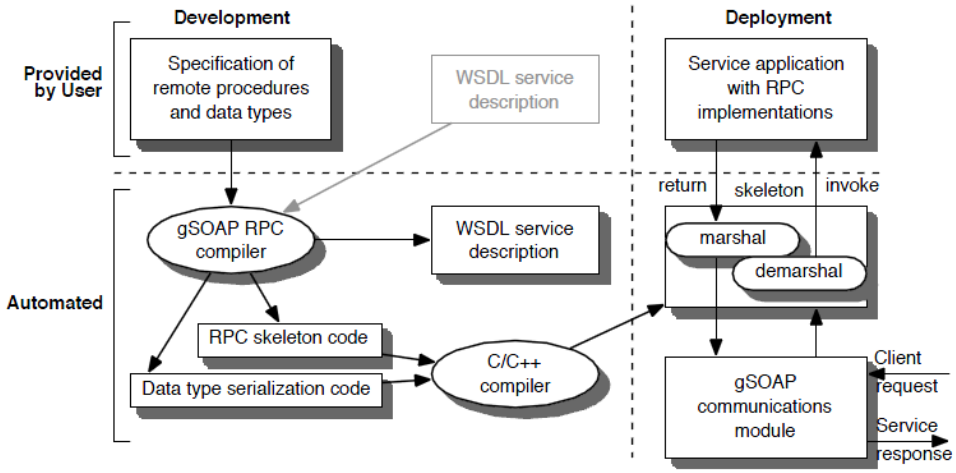

Figura 24. Desenvolvimento e exploração de um serviço

É frequente encontrarem-se redes com equipamentos com capacidades muito díspares, como *Smart Grids* ou redes de controlo industrial, que podem incluir desde simples sensores até poderosos servidores de agregação. Para suportar dispositivos com recursos limitados, o IETF lançou um *Internet-Draft* que documenta um modo de operação mais "leve", o *NETCONF Light*[140]. As implementações desta versão não suportam as capacidades extra definidas na RFC 4741 e podem suportar um número muito limitado de sessões, podendo mesmo ser só uma, e só utiliza o repositório *running* que é o mínimo exigido pelo protocolo. No que diz respeito às operações, são obrigatórias as operações *get*, *get-config*, *copy-config*, *lock*, *unlock*, *close-session* e *kill-session*. As operações *edit-config* e *delete-config* são opcionais, sendo que a última deixa mesmo de fazer sentido uma vez que só existe um repositório de dados. A opção de filtragem também é opcional, passando-se a manipular as configurações no seu todo, o que não é crítico uma vez que tipicamente serão configurações reduzidas. Embora o *NETCONF light* seja um subconjunto de funcionalidades do protocolo base, também deve ser anunciado como uma capacidade através do URI:

urn:ietf:params:netconf:light:1.1

Algo também essencial para este projeto foi a implementação de um módulo de monitorização com notificações assíncronas de eventos baseado na RFC 5277 [26]. O gSOAP só permite dois tipos de interfaces de comunicação: uma *request-response*, onde o servidor recebe um pedido e gera uma resposta para o cliente, e outra *one-way*, em que o servidor recebe mensagens mas não gera respostas. Assim, atendendo a que as funções de cliente e servidor se alteram durante a produção normal do NETCONF e entre as notificações de eventos, foi necessário desenvolver dois *web service*s. Um é um agente de NETCONF que recebe pedidos dos gestores e responde adequadamente e o outro é um *listener* de notificações de eventos que corre num gestor após um processo de subscrição de eventos bem sucedido. Este processo é visível na **Figura 25**, em que o cliente e o servidor de notificações são executados em *threads*, respetivamente do servidor e do cliente de NETCONF.

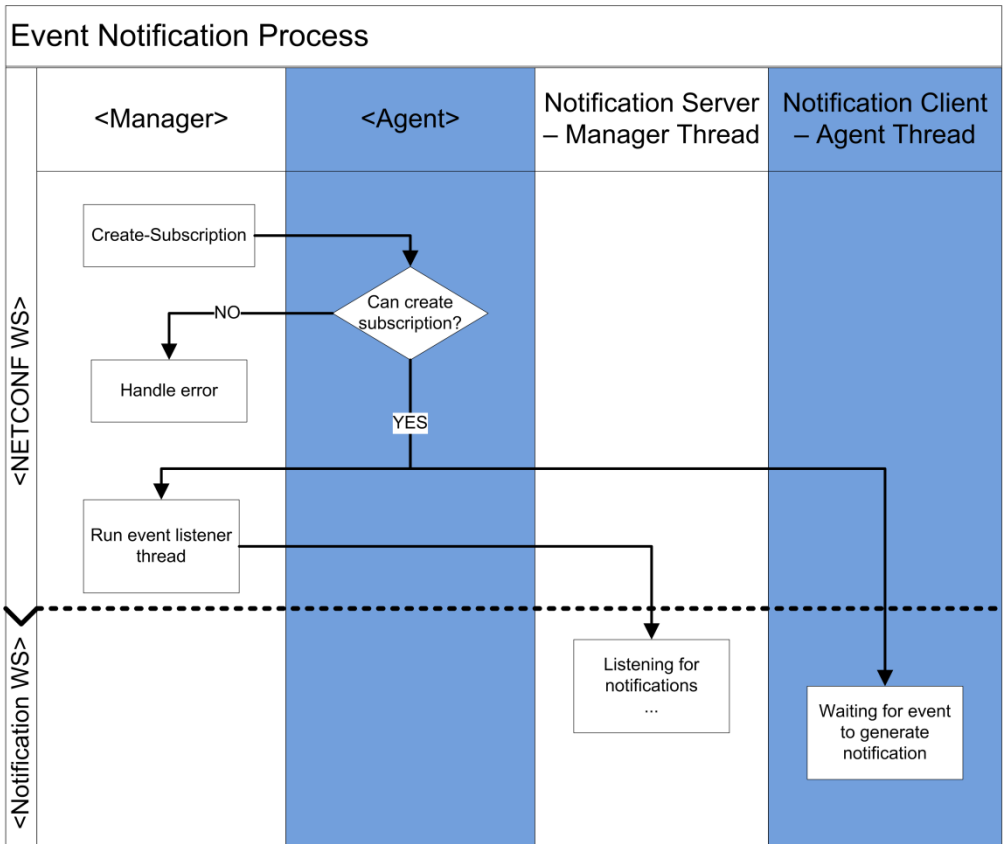

Figura 25. Processo de Notificação de Eventos

3.2. Modelo de dados

Como modelos de dados para o *web service* de NETCONF usaram-se os modelos propostos pelo IETF. Na RFC 4743, que define o documento WSDL com uma definição das operações do *web service*, os tipos de dados e as operações do NETCONF são definidas num *XML schema* presente na RFC 4741. De forma análoga, na RFC 5277 é definido o *XML schema* das operações de subscrição e notificação de eventos a integrar na arquitetura NETCONF. Na **Figura 26** mostra-se um diagrama de uma mensagem <*rpc-request*> conforme a definição do modelo de dados. Os documentos do modelo de dados constituem os anexos de A a D.

Para este projeto integrou-se a operação de subscrição no *XML schema* do NETCONF e criou-se um novo WSDL para definir o *web service* de notificação que tem a definição da mensagem de notificação assíncrona de eventos.

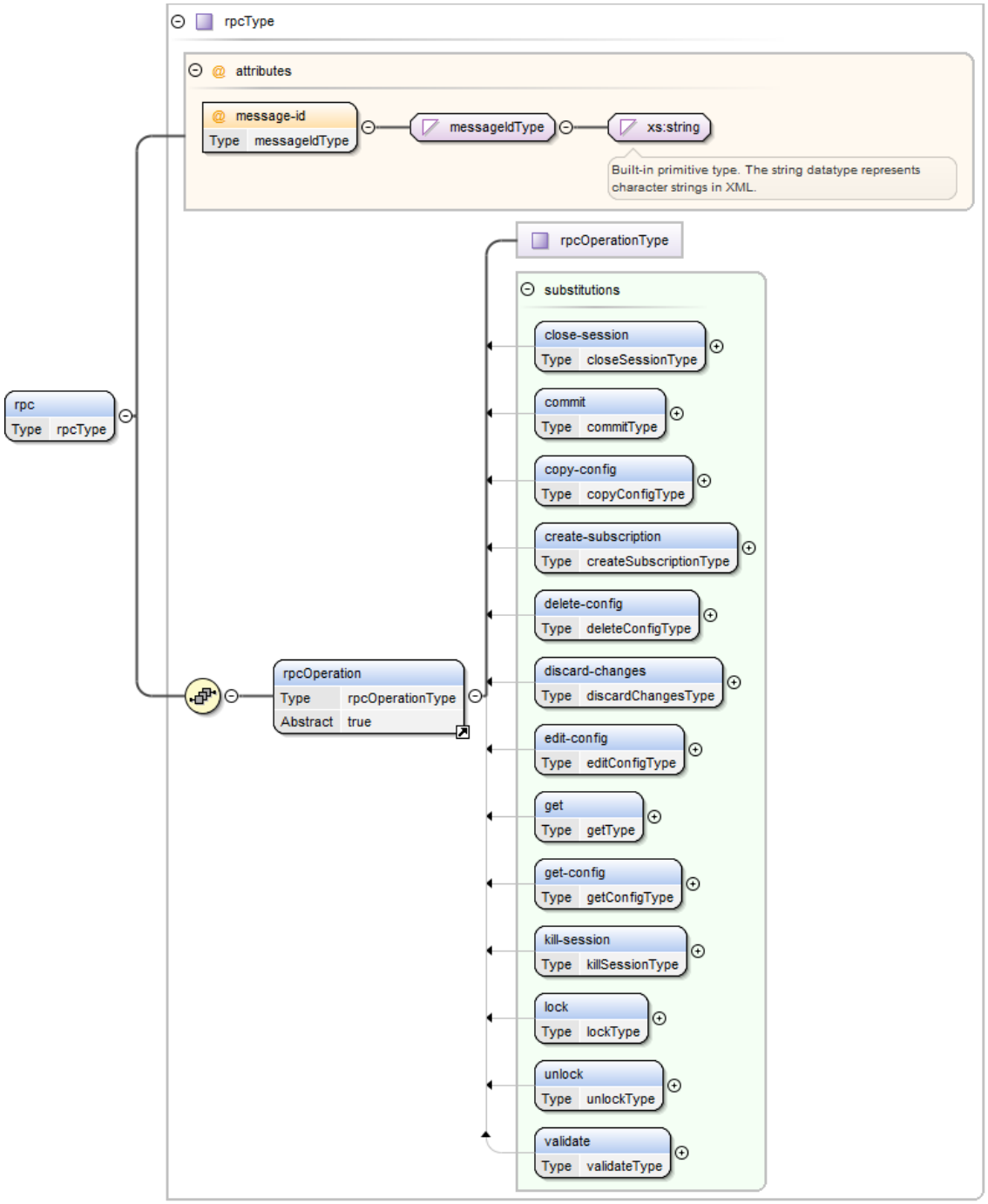

Figura 26. Diagrama de um <*rpc-request*> conforme definido no modelo de dados

3.3. Mensagens trocadas

Nesta secção pretende-se descrever as mensagens trocadas para iniciar e terminar uma sessão NETCONF, bem como as mensagens das principais operações.

Numa implementação de NETCONF sobre SOAP/HTTP é usado um cabeçalho HTTP para transmitir uma mensagem SOAP, que por sua vez contém um elemento <*hello*>, um elemento <*rpc*> ou um elemento <*rpc-reply*>. O elemento <*hello*> é utilizado para iniciar uma sessão NETCONF, o elemento <*rpc*> é utilizado pelo gestor de NETCONF para enviar operações para um agente, que por sua vez, utiliza o elemento <*rpc-reply*> para responder ao agente.

Assim, para iniciar uma sessão um cliente envia uma mensagem <*hello*> com as capacidades que suporta, **Caixa de Código 1**.

```
C: POST / HTTP/1.1
C: Host: localhost
C: User-Agent: gSOAP/2.7
C: Content-Type: text/xml; charset=utf-8
C: Content-Length: 753
C: Connection: keep-alive
C: SOAPAction: ""
C:
C: <?xml version="1.0" encoding="UTF-8"?>
C: <SOAP-ENV:Envelope
C:     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
C:     xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
C:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
C:     xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
C:     xmlns:ns3="urn:ietf:params:xml:ns:netconf:soap:1.0"
C:     xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0">
C:     <SOAP-ENV:Body>
C:            <ns1:hello>
C:                   <ns1:capabilities>
C:                          <ns1:capability>
C:                                 urn:ietf:params:xml:ns:netconf:base:1.0
C:                          </ns1:capability>
C:                          <ns1:capability>
C:                                 urn:ietf:params:netconf:light:1.1
C:                          </ns1:capability>
C:                          <ns1:capability>
C:                                 urn:ietf:params:netconf:soap:1.1
C:                          </ns1:capability>
C:                   </ns1:capabilities>
C:            </ns1:hello>
C:     </SOAP-ENV:Body>
C: </SOAP-ENV:Envelope>
```

Caixa de Código 1. Mensagem <hello> de resposta enviada por um cliente

Em resposta deve vir uma mensagem <*hello*> com as capacidades do servidor, **Caixa de Código 2**. São então escolhidas as versões mais recentes das capacidades comuns para serem utilizadas durante a sessão. O servidor também deve incluir na resposta um número identificador da sessão, <*session-id*>.

```
S: HTTP/1.1 200 OK
S: Server: gSOAP/2.7
S: Content-Type: text/xml; charset=utf-8
S: Content-Length: 820
S: Connection: keep-alive
S:
S: <?xml version="1.0" encoding="UTF-8"?>
S: <SOAP-ENV:Envelope
S:     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
S:     xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
S:     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
S:     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
S:     xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
S:     xmlns:ns3="urn:ietf:params:xml:ns:netconf:soap:1.0"
S:     xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0">
S:     <SOAP-ENV:Body>
S:            <ns1:hello>
S:                   <ns1:capabilities>
S:                          <ns1:capability>
S:                          </ns1:capability>
S:                          <ns1:capability>
S:
       urn:ietf:params:xml:ns:netconf:base:1.0
S:                          </ns1:capability>
S:                          <ns1:capability>
S:                                 urn:ietf:params:netconf:light:1.1
S:                          </ns1:capability>
S:                          <ns1:capability>
S:                                 urn:ietf:params:netconf:soap:1.1
S:                          </ns1:capability>
S:                   </ns1:capabilities>
S:                   <ns1:session-id>1</ns1:session-id>
S:            </ns1:hello>
S:     </SOAP-ENV:Body>
S: </SOAP-ENV:Envelope>
```

Caixa de Código 2. Mensagem <hello> de resposta enviada por um servidor

As mensagens <*rpc*> e <*rpc-reply*> devem conter um valor identificador, <*message-id*>, que permite a ambos fazer a correspondência entre um pedido e uma resposta. Encapsulados num <*rpc*> seguem como elementos RPC os parâmetros da operação.

Na **Caixa de Código 3** está ilustrada uma mensagem <*rpc*> com uma operação <*get-config*> que permite pedir os dados de configuração (sem os dados de estado) a um agente. Neste caso pede-se as configurações do repositório *running*.

```
C: <ns1:rpc xsi:type="ns1:rpcType" message-id="1">
C:     <ns1:get-config xsi:type="ns1:getConfigType">
C:           <ns1:source xsi:type="ns1:getConfigSourceType">
C:                 <running xsi:type="ns1:configNameType"
C:                 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
C:                 </running>
C:           </ns1:source>
C:           <filter xsi:type="ns1:filterInlineType"
C:                 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
C:                 type="subtree">
C:           </filter>
C:     </ns1:get-config>
C: </ns1:rpc>
```

Caixa de Código 3. Mensagem <rpc> com uma operação <get-config>

Caso o pedido possa ser satisfeito o servidor envia um <*rpc-reply*> com o mesmo *message-id*, um elemento <*ok*> e os dados de resposta, **Caixa de Código 4**.

```
S: <ns1:rpc-reply xsi:type="ns1:rpcReplyType" message-id="1">
S:     <data xsi:type="ns1:dataInlineType"
S:           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
S:           ...
S:     </data>
S:     <ok xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
S:     </ok>
S: </ns1:rpc-reply>
```

Caixa de Código 4. Mansegem <rpc-reply> de uma operação <get-config> bem sucedida

Em caso de erro o <*rpc-reply*> inclui um ou mais elementos <*rpc-error*> que contêm uma descrição do erro, **Caixa de Código 5**. Essa discrição é feita por elementos como *error-type*, *error-tag*, *error-severity*, *error-message* ou *error-info*. O elemento error-type serve para indicar a camada protocolar onde o erro se verifica e pode tomar valores como *transport*, *rpc*, *protocol* ou *aplication*. O *error-tag* usa-se para identificar o erro em si, para o qual a RFC 4741 disponibiliza uma lista de erros possíveis onde constam erros como *invalid-value* ou *missing-attribute*. O *error-severity* indica se é um aviso (*warning*) ou um erro (*error*). Utiliza-se o *error-message* para transportar uma *string* que possa ser apresentada ao utilizador para descrição do erro. O elemento *error-info* descreve informação específica do protocolo ou modelo de dados, a RFC 4741 também define o conteúdo obrigatório deste elemento perante alguns erros. Existe também o elemento *error-app-tag*, para descrever erros relacionados com a implementação, ou o elemento *error-path* que pode ser utilizado para identificar o elemento relacionado com o erro. Estes elementos não são todos obrigatórios e devem ser usados de acordo com a situação em causa. A RFC normaliza uma lista de erros e descrições a utilizar, mas dá liberdade para se adicionarem novas descrições ou mesmo novos elementos para descrever os erros.

```
S: <ns1:rpc-reply xsi:type="ns1:rpcReplyType" message-id="1">
S:     <ns1:rpc-error xsi:type="ns1:rpcErrorType">
S:           <ns1:error-type> application </ns1:error-type>
S:           <ns1:error-tag> data-missing </ns1:error-tag>
S:           <ns1:error-severity> error </ns1:error-severity>
S:           <ns1:error-info xsi:type="ns1:errorInfoType">
S:           </ns1:error-info>
S:     </ns1:rpc-error>
S:     <ok xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
S:     </ok>
S: </ns1:rpc-reply>
```

Caixa de Código 5. Mensagem <rpc-error> enviada por um servidor

Não menos importante e essencial ao bom funcionamento do protocolo é o terminar de uma sessão de NETCONF, representado na **Caixa de Código 6**. Para por termo a uma sessão um cliente envia um <*close-session*> que deve ser sucedido de uma mensagem com um elemento <*ok*> por parte do servidor. O fim da sessão implica que sejam libertados os recursos que lhe estão alocados, como *locks*, sessão TCP ou subscrições de notificações.

```
C: <ns1:rpc xsi:type="ns1:rpcType" message-id="1">
C:     <close-session xsi:type="ns1:closeSessionType"
C:                    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
C:     </close-session>
C: </ns1:rpc>

S: <ns1:rpc-reply xsi:type="ns1:rpcReplyType" message-id="1">
S:     <ok xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
S:     </ok>
S: </ns1:rpc-reply>
```

Caixa de Código 6. Troca de mensagens para terminar uma sessão de NETCONF

Também relevante para este trabalho é a subscrição de eventos (**Caixa de Código 7**) e o respetivo envio de notificações (**Caixa de Código 8**). Assim, para subscrever notificações de eventos um cliente envia um elemento <*create-subscription*> e, em caso afirmativo, o servidor responde com um elemento <*ok*>, numa mensagem <*rpc-reply*>. Após esta troca de mensagens o servidor envia notificações sempre que se verifiquem eventos que o justifiquem. Este processo está ilustrado na **Figura 27**.

```
C: <ns1:rpc xsi:type="ns1:rpcType" message-id="1">
C:     <ns1:create-subscription xsi:type="ns1:createSubscriptionType">
C:           <ns1:stream></ns1:stream>
C:     </ns1:create-subscription>
C: </ns1:rpc>
```

Caixa de Código 7. Mensagem para criar uma subscrição de eventos em NETCONF

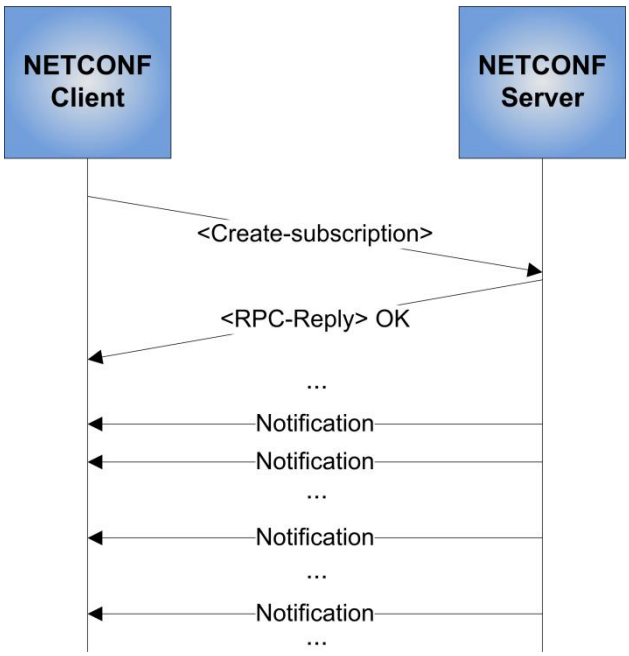

Figura 27. Troca de mensagens para a monitorização por notificação de eventos do NETCONF

Após uma subscrição de notificações de eventos válida um servidor envia as notificações de forma assíncrona, sempre que se verifiquem eventos respeitantes à *stream* subscrita e ao critério de filtragem, caso exista. De notar que uma notificação não é uma operação RPC do NETCONF, mas sim um elemento de alto nível que identifica uma mensagem unidirecional que representa uma notificação de eventos. Nas notificações é obrigatório ter um selo temporal de quando o evento foi gerado na sua origem. O NETCONF permite que se adicione informação extra acerca do evento mas não a normaliza, ficando, para já, ao critério da implementação.

```
<ns2:notification xsi:type="ns2:NotificationType">
      <ns2:eventTime>
            2011-09-15T11:45:15Z
      </ns2:eventTime>
</ns2:notification>
```

Caixa de Código 8. Notificação de um evento

4. Testes e análise de resultados

Nesta secção descrevem-se os testes efetuados e analisam-se os resultados obtidos. São executados testes para analisar o tráfego gerado, a eficiência de sinalização e os tempos de resposta dos diferentes protocolos.

4.1. Comparação com outras Tecnologias

Após a análise de vários protocolos para monitorização de redes IP, é de grande interesse comparar o desempenho das diferentes abordagens. Para isso, decidiu-se confrontar o *net-snmp* [141], uma implementação de SNMP desenvolvida na universidade de *Carnegie Mellon*, a implementação do WBEM *openpegasus* e a implementação de NETCONF feita ao abrigo da presente dissertação. Todas as implementações são escritas em C/C++, o que lhes confere uma base de partida comum, não havendo diferenças substanciais a este nível.

Para comparar o desempenho destas implementações decidiu-se analisar a eficiência de sinalização nas operações de monitorização, o tráfego gerado comparando o número de pacotes e o número de bytes transmitidos e as respostas dos protocolos quando submetidos a testes de carga.

Os testes foram executados numa máquina com um processador Intel Mobile Core 2 Duo @ 2.20GHz, 2GB de RAM numa instalação nativa de Ubuntu 10.10. Para a recolha de dados recorreu-se ao Wireshark 1.6.2 e a *bash scripting* para geração intensiva de notificações de eventos.

No caso do WBEM, para criar uma subscrição de indicações foi necessário criar instâncias das classes *CIM_IndicationFilter*, *CIM_ListenerDestinationCIMXML* e *CIM_IndicationSubscribe*. Para isso usou-se o comando *cimcli* da plataforma *OpenPegasus*, como ilustrado na **Caixa de Código 9**.

```
### Criar Filtro (CIM_IndicationFilter)

$ cimcli ci -n SDKExamples/DefaultCXX CIM_IndicationFilter
CreationClassName=CIM_IndicationFilter Name=TestFilter
SourceNamespace="SDKExamples/DefaultCXX" Query="SELECT * FROM
RT_TestIndication" QueryLanguage=WQL

### Criar Handler (CIM_ListenerDestinationCIMXML)

$ cimcli ci -n SDKExamples/DefaultCXX CIM_ListenerDestinationCIMXML
CreationClassName=CIM_ListenerDestinationCIMXML Name=TestListener
Destination=http://localhost:5999/CIMListener/SimpleDisplayConsumer

### Criar Subscrição (CIM_IndicationSubscription)

cimcli ci -n root/PG_InterOp CIM_IndicationSubscription
Filter=SDKExamples/DefaultCXX:CIM_IndicationFilter.CreationClassName=\"
CIM_IndicationFilter\",Name=\"TestFilter\",SystemCreationClassName=\"CI
M_ComputerSystem\",SystemName=\"ubuntu\"
Handler=SDKExamples/DefaultCXX:CIM_ListenerDestinationCIMXML.CreationCl
assName=\"CIM_ListenerDestinationCIMXML\",Name=\"TestListener\",SystemC
reationClassName=\"CIM_ComputerSystem\",SystemName=\"ubuntu\"
```

Caixa de Código 9. Registo de subscrição de indicações WBEM usando o *cimcli*

Usou-se um *indication provider*, o *RT_Indication provider*, fornecido com o OpenPegasus para teste de indicações, no CIMOM *cimserver*, foi registada uma versão adaptada deste *provider* para emitir indicações com a informação pretendida neste teste. Assim, para gerar indicações é necessário invocar o método *SendTestIndication* desse *provider*. Para executar esse método o OpenPegasus fornece uma aplicação com o mesmo nome. Adicionalmente, é necessário executar um *deamon* para receber as indicações, o *cimlistener*. Para gerar fluxos de indicações criou-se um *script* (**Caixa de Código 10**) que invoca consecutivamente a aplicação que provoca o envio de indicações.

```
#!/bin/bash

if [ -z $1 ]; then
        ITERACOES=10
else
        ITERACOES=$1
fi

for ((i=0; i<$ITERACOES; i++))
do
        sudo ./SendTestIndications >/dev/null
done
```

Caixa de Código 10. Script para gerar indicações do WBEM

Para testar o SNMP usou-se o conjunto de aplicações Net-SNMP, o comando *snmptrap* foi utilizado para gerar as notificações de eventos e a aplicação *snmptrapd* para receber as *traps*.

```
$ snmptrap -v 2c -c public 127.0.0.1 '' IF-MIB::linkDown IF-
MIB::ifIndex i 1 IF-MIB::ifAdminStatus i 1 IF-MIB::ifOperStatus i 1
```

Caixa de Código 11. Comando para gerar as *traps* do teste

O comando descrito na **Caixa de Código 11** envia uma *trap* da segunda versão do SNMP, com a informação *link down* acerca de uma interface identificada por *ifIndex*. Para os testes criou-se um *script* idêntico ao usado no WBEM para enviar várias *traps*.

No caso do NETCONF, as aplicações cliente e servidor foram adaptadas para serem usadas nestes testes. As aplicações NETCONF do teste não trocam as mensagens <*Hello*> para iniciar uma sessão NETCONF; quando é iniciada, a aplicação cliente fica à escuta de notificações, enquanto a aplicação servidor quando é iniciada estabelece a sessão TCP e envia o número de notificações recebido pela linha de comandos. Assim, foi possível usar um *script* idêntico aos usados nos restantes protocolos para enviar fluxos de notificações.

4.2. Análise de tráfego

A análise de tráfego prende-se com a necessidade de entender a influência que a monitorização pode ter no desempenho de uma rede de dados IP, comparando o desempenho dos diferentes protocolos ao nível das transferências de dados.

Nesta análise recolheram-se dados relativos à quantidade de pacotes (**Figura 28**) e ao número de bytes transmitidos (**Figura 29**) durante o processo de monitorização com notificações de eventos.

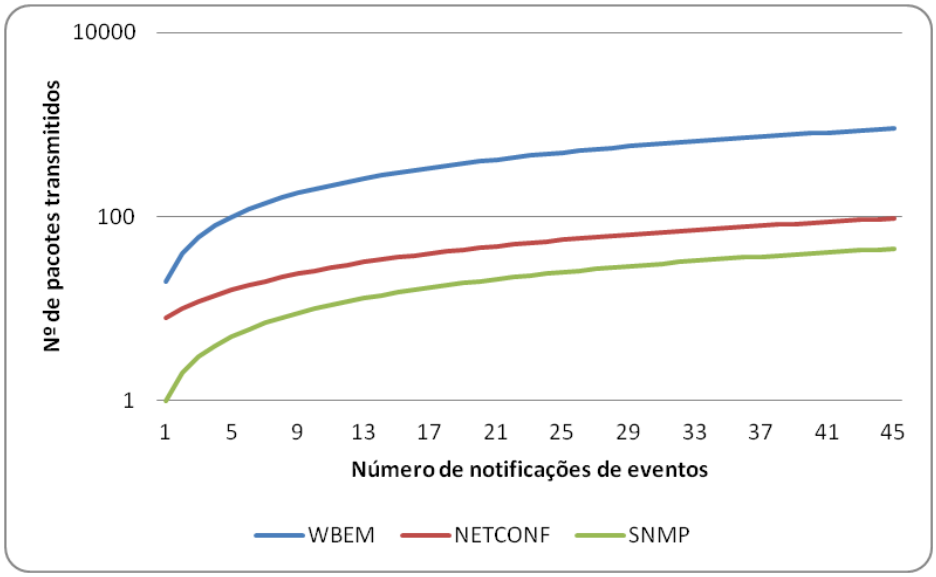

Figura 28. Gráfico do número de pacotes transmitidos

Assim, verificou-se que o WBEM troca um elevado número de pacotes quando comparado com os outros protocolos, sendo o número de bytes transmitidos também o maior valor registado de entre todos os protocolos. No WBEM cada indicação é transmitida numa sessão TCP diferente, o que impõe um *overhead* que não é tão significativo no NETCONF, uma vez que é criada uma sessão TCP por cada subscrição/*stream* de eventos. Em consequência, o NETCONF transmite em média 13% menos pacotes que o WBEM. No SNMP, que recorre ao protocolo de transporte UDP, este *overhead* não se verifica de todo uma vez que não existe uma sessão ao nível da camada de transporte. Então, o envio de um *trap* implica a transmissão de um único pacote de dados enquanto no NETCONF são trocados dois pacotes, uma vez que a receção de um pacote é sempre confirmada pelo destino através de uma mensagem de *acknowledge*. Este facto faz com que o SNMP envie em média 40% menos pacotes que o NETCONF.

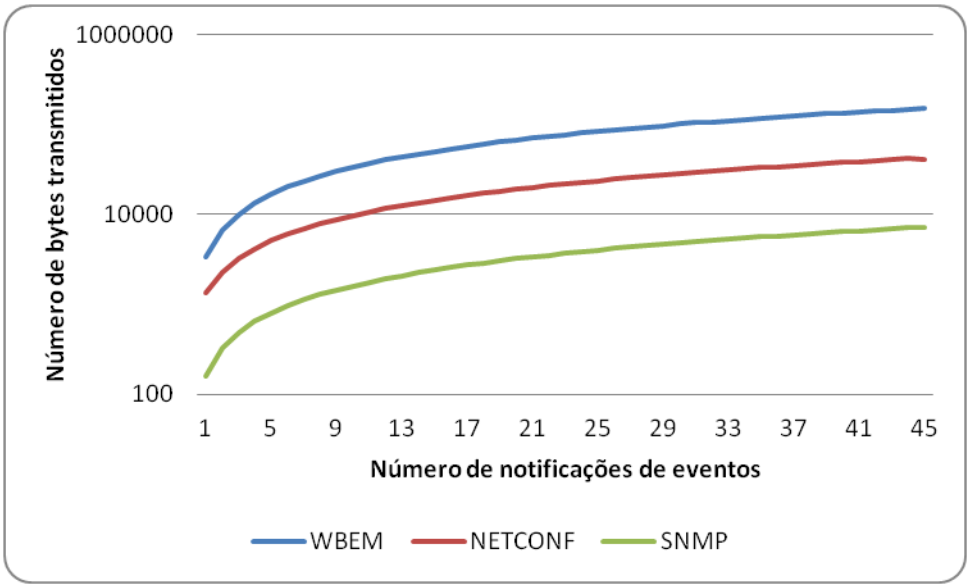

Figura 29. Gráfico do número de bytes transmitidos

O WBEM, por criar e terminar uma sessão TCP para cada indicação e por causa da elevada quantidade de dados transmitidos e respetivas confirmações de receção, troca vinte mensagens por indicação, um valor bastante superior ao do NETCONF, que transmite em média 29% menos bytes que o WBEM.

Quando se compara o número de bytes transmitidos (**Figura 29**) nota-se uma grande disparidade entre os protocolos transmitidos sobre TCP e com codificação XML, mais verbosa, e o SNMP que é transmitido sobre UDP e codificado em binário. O funcionamento dos primeiros requer a transmissão de grandes quantidades de bytes, enquanto o SNMP utiliza em média menos 16.6% de bytes que o NETCONF, que apresenta os melhores resultados do primeiro grupo de protocolos.

4.3. Análise da Eficiência de Sinalização

Para se comparar a eficiência de sinalização de cada uma das tecnologias mediu-se a sinalização e calculou-se a eficiência dos diferentes protocolos. Nos testes foram usadas notificações de eventos que indicassem uma alteração do estado de uma interface. Este é um acontecimento bastante comum e passível de acontecer, sendo normalmente identificado por descrições como *"Link Up"* e *"Link Down"*.

Tipicamente, as notificações de eventos têm um campo com uma descrição do evento que gerou a notificação e uma referência temporal de quando o mesmo ocorreu. De seguida apresenta-se a indicação do WBEM referente à notificação do evento de teste.

```
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
<MESSAGE ID="1010" PROTOCOLVERSION="1.0">
<SIMPLEEXPREQ>
<EXPMETHODCALL NAME="ExportIndication">
<EXPPARAMVALUE NAME="NewIndication">
<INSTANCE CLASSNAME="RT_TestIndication" >
<PROPERTY NAME="IndicationIdentifier"  TYPE="string">
<VALUE>63486176495792572</VALUE>
</PROPERTY>
<PROPERTY NAME="IndicationTime"  TYPE="datetime">
<VALUE>20111018180135.792583+060</VALUE>
</PROPERTY>
<PROPERTY.ARRAY NAME="CorrelatedIndications"
TYPE="string">
<VALUE.ARRAY>
</VALUE.ARRAY>
</PROPERTY.ARRAY>
<PROPERTY NAME="IndicationDescription"  TYPE="string">
<VALUE>eth0: link-down</VALUE>
</PROPERTY>
<PROPERTY NAME="MethodName"  TYPE="string">
<VALUE>generateIndication</VALUE>
</PROPERTY>
</INSTANCE>
</EXPPARAMVALUE>
</EXPMETHODCALL>
</SIMPLEEXPREQ>
</MESSAGE>
</CIM>
```

Caixa de Código 12. CIM-XML da indicação WBEM do teste

O CIM, por causa da sua arquitetura orientada a objetos, inclui *tags* para identificar elementos como classes, instâncias, propriedades ou valor. A presença destas *tags* aumenta substancialmente a quantidade de informação a transmitir, penalizando bastante a eficiência da sinalização da informação. Já a notificação de NETCONF usada no teste (**Caixa de Código 13**), mesmo sendo codificada em XML, recorre a muito menos *tags* beneficiando bastante a quantidade de informação transferida em cada notificação.

```
<ns2:notification
xsi:type="ns2:NotificationType">
       <ns2:eventTime>
              2011-10-17T17:16:27Z
       </ns2:eventTime>
       <ns2:info>
              link up
       </ns2:info>
</ns2:notification>
```

Caixa de Código 13. XML da notificação de teste do NETCONF

As traps de SNMP utilizadas no teste, representadas na **Figura 30,** indicam a alteração do estado de uma interface de rede. Para isso, contêm um OID indicativo de quando ocorreu o evento (UpTime), um OID referente a Link-Down e outro referente ao identificador da interface.

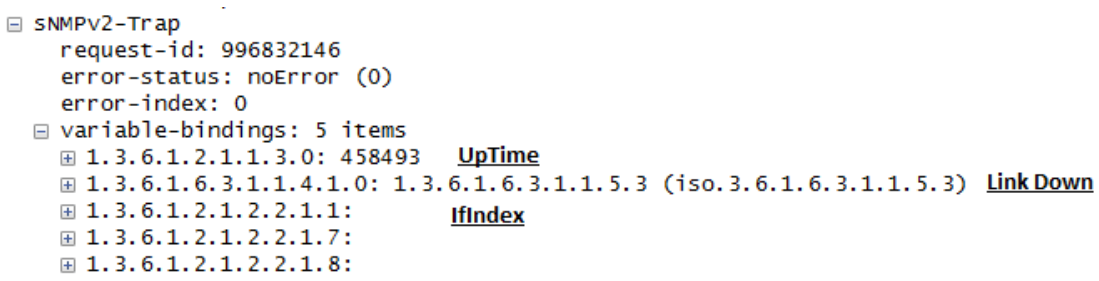

Figura 30. Trap de SNMP utilizada nos testes

Na **Tabela 11** apresentam-se os valores da sinalização total e da sinalização referente aos dados pertinentes para a notificação de eventos, em cada protocolo.

Tabela 11. Dados referentes à sinalização de dados nos diferentes protocolos (bytes)

	WBEM	*NETCONF*	*SNMP*
Descrição do evento	15	7	25
Referência temporal	25	20	17
Total Inf. Pertinente	40	27	42
Sinalização Total	3374	1347	161
Eficiência	1.19%	2.00%	26.09%

Nesta tabela pode-se verificar uma diferença notável entre o uso de codificação binária ou XML. A codificação binária não usa *tags* e é uma forma mais eficiente de codificação: por exemplo, o tipo de dados *datetime* usado no WBEM é codificado recorrendo a 8 bytes, mas quando transmitido em XML ocupa 25 bytes.

4.4. Análise de tempos de resposta

Para a análise dos tempos de resposta adicionou-se o comando *time* do *Linux* aos scripts mencionados nas secções anteriores, usando a assinatura da **Caixa de Código 14**. Este comando mede os recursos de sistema utilizados por um processo, neste caso usou-se especificamente para medir o tempo de execução de um processo.

```
$ time –a –o $LOG –f  "ElapsedTime:%E" ./testApp
```

Caixa de Código 14. Assinatura do comando *time* usada nos testes

Assim, os resultados obtidos na análise dos tempos de resposta seguem o padrão esperado e visível na **Figura 31**. O SNMP tem um desempenho superior ao WBEM e ao NETCONF; esta vantagem deve-se ao uso do UDP na camada de transporte e à baixa quantidade de dados a transmitir. Entre os protocolos que recorrem ao TCP, o NETCONF tem o melhor desempenho, consequência do uso do YANG para descrição de dados e operações. A codificação CIM-XML utilizada pelo WBEM é muito verbosa, necessitando de várias *tags* nas suas descrições dos dados, pelo que este facto penaliza em muito o desempenho do WBEM.

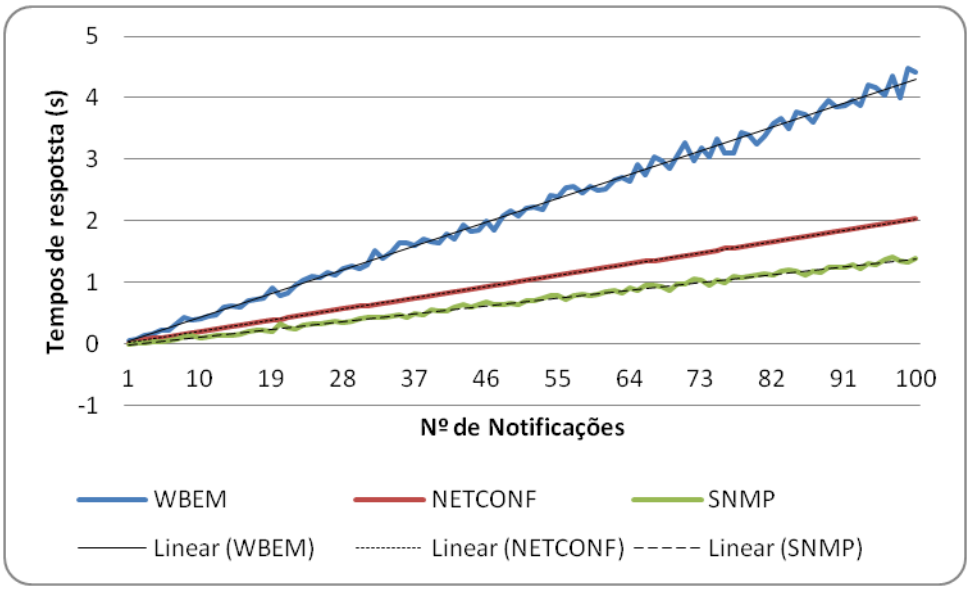

Figura 31. Gráfico dos testes dos tempos de resposta

A irregularidade nos gráficos da **Figura 31** deve-se a algum ruído durante a aquisição de dados. Este ruído deve-se a processos que eram escalonados pelo sistema operativo e que estando a ser executados de forma concorrente atrasavam a execução do processo em estudo. O ruído é maior no WBEM pois estão a ser medidas as indicações enviadas pelo *CIMServer* para o *CIMListener* enquanto que concorrentemente estão a ser feitos pedidos pela aplicação *SendTestIndication* ao *CIMServer*, enquanto nos outros testes

se usou apenas uma aplicação para gerar notificações de eventos e um *listener* para as receber e interpretar. Mas mesmo assim, a análise dos gráficos dos valores médios permite concluir que estes tempos crescem de forma linear e que os protocolos apresentam tempos de resposta distintos, como também se pode concluir pela análise da **Tabela 12**.

Tabela 12. Tabela dos tempos médios de resposta dos diferentes protocolos

	WBEM	*NETCONF*	*SNMP*
Média do tempo de resposta (s)	0.043	0.0203	0.0129

Assim, verifica-se que em média o NETCONF necessita de sensivelmente 47% do tempo de resposta do WBEM, enquanto o SNMP necessita apenas de 30%.

5. Conclusões e trabalho futuro

Desde meados dos anos 80 que têm surgido inúmeras tecnologias para gestão e monitorização de redes, mas até hoje nenhum gerou consenso na satisfação desse objetivo, quer por problemas de complexidade, de interoperabilidade ou mesmo problemas relacionados com o mercado. Assim, em 2006 apareceu uma nova iniciativa, o NETCONF, cujas premissas tiraram partido da larga experiência e conhecimento do sector, tendo envolvido desde a sua criação entidades como o IETF, *CISCO Systems*, *Juniper Networks*, entre outros, que puseram ao seu dispor alguns dos profissionais mais influentes da área. Este é um protocolo recente, que surgiu como uma nova esperança para a gestão e monitorização de redes IP. Assim, neste trabalho tentou-se entender o que o NETCONF trouxe de novo ou de diferenciador das tecnologias existentes e quais as reais consequências dessas inovações. Por fim, tentou-se estudar em detalhe as capacidades de monitorização implementadas através de notificações de eventos, que por razões já descritas é uma ferramenta chave para uma boa prestação de serviços duma rede ou sistema.

Durante muito tempo o SNMP foi um recurso valioso dos administradores de redes, assumindo-se como a tecnologia de gestão *de facto*, apesar de a sua utilização estar tipicamente restrita às ações de monitorização graças a um conjunto de defeitos que desde cedo lhe foram apontados. Estes defeitos vão desde a elevada complexidade de implementação das MIBs, também consequência da utilização do SMI, cujo modelo relacional não se adequa devidamente à descrição da informação de gestão; problemas ao lidar com grandes transferências de informação devido ao uso do UDP na camada de transporte, que não oferece nenhum tipo de garantia de receção, o que pode trazer problemas relativos à consistência das comunicações; o fraco empenhamento do mercado em prestar suporte a estas soluções e em fornecer atempadamente modelos de dados atualizados; problemas relacionados com segurança, que só tiveram solução no SNMPv3 e mesmo assim tinham problemas de interoperabilidade com soluções de autenticação já existentes. Assim, o SNMP nunca chegou a ser devidamente adotado como uma solução de configuração, sendo utilizado principalmente para as funções de monitorização.

Um dos esforços mais relevantes para a gestão e monitorização de redes e sistemas é o WBEM, do DMTF, que também conta com a participação da *CISCO Systems* e da *Juniper Networks*, bem como a *Huawei*, *IBM* ou *Intel*, de entre uma lista enorme de *players* influentes do sector. O WBEM baseia a sua inovação numa nova linguagem para descrição da informação de gestão, o CIM. O CIM segue uma arquitetura orientada a objetos que se afigurava mais indicada para descrever a informação de gestão e monitorização. A par da nova linguagem para descrever os dados, o WBEM usa o XML para codificar os dados de gestão, mais vocacionado para uma descrição hierárquica dos dados. O WBEM segue ainda a tendência baseada em Web dos sistemas de informação, uma tendência cada vez mais forte e que se faz notar nas mais variadas e recentes soluções tecnológicas. Assim, o WBEM normaliza uma solução de gestão e monitorização sobre HTTP cujos objetivos principais são facilitar a gestão em ambientes distribuídos e a interoperabilidade entre diferentes plataformas.

Mais tarde, e seguindo a tendência baseada na Web e com o aparecimento das arquiteturas orientadas a serviços, surgem normalizações de gestão sobre web-services, nomeadamente o WS-MAN do DMTF e o MUWS da OASIS. Ambas as soluções

pretendem usar as definições do W3C para web-services, para normalizar soluções unificadas e interoperáveis que facilitem o acesso remoto à informação de gestão de recursos distribuídos numa rede.

Mais recentemente surgiu o NETCONF direcionado para gestão e monitorização de configurações em dispositivos de rede. O NETCONF tenta dar resposta a vários problemas da gestão de redes identificados ao longo do tempo, desenvolvendo uma nova linguagem para descrição dos dados e operações de gestão, o YANG. O protocolo conta com várias inovações: recorre ao XML para codificar os dados, o que permite o *parsing* usando inúmeras implementações já existentes, não necessitando de ferramentas específicas; permite a distinção entre dados de estado e de configuração; prevê a adaptação dos modelos de dados para permitir a descrição dos dados de diferentes equipamentos recorrendo a um único protocolo. Permite ainda, a definição de novas operações promovendo a interoperabilidade entre diferentes dispositivos ou versões, permite a integração com métodos existentes de autenticação, suporta diversos repositórios de dados, permitindo a sua manipulação total ou parcial; suporta a transação de configurações com operações de *lock* e *rollback*. Uma das suas grandes características é o suporte de vários protocolos de transporte com segurança, sendo que o SSH é obrigatório, está também previsto o uso de BEEP, TLS ou SOAP. Também está definido o suporte de monitorização através de notificações assíncronas de eventos. Este conjunto de inovações torna o NETCONF único mesmo quando comparado com o WBEM, provavelmente o seu concorrente natural.

Ao abrigo desta dissertação fez-se uma implementação do protocolo NETCONF, especificamente da sua versão *light*, escolhendo SOAP para o transporte das mensagens RPC. O *NETCONF-light* é uma versão do protocolo que permite a sua implementação em dispositivos com baixos recursos, o que aumenta substancialmente a sua aplicabilidade nos diferentes tipos de dispositivos que se podem encontrar numa rede de comunicações. Esta versão do NETCONF diminui os requisitos principalmente no que toca à funcionalidade de filtragem, que interfere bastante com a capacidade de processamento e de memória. O uso de SOAP deve-se à tendência que se tem vindo a notar nos últimos anos para o aumento do número de serviços com interfaces Web e implementações que seguem o paradigma orientado a serviços. Nesta implementação, incluíram-se as capacidades de notificação de eventos, permitindo posteriormente comparar os seus requisitos e desempenho com outros protocolos.

Após a análise das características dos diferentes protocolos, das suas vantagens e desvantagens procedeu-se a uma comparação prática sobre as capacidades destes protocolos. Esta comparação prática fez-se através de vários testes que, mesmo sendo executados num ambiente controlado, em laboratório, oferecem uma ideia do que se pode esperar da eficiência destes protocolos quando postos em ambientes de produção. Executaram-se testes que permitiram medir a eficiência da sinalização de cada protocolo, contabilizar o número de pacotes e de bytes trocados, permitindo tirar elações sobre as consequências da sua aplicação para o tráfego de uma rede de dados e ainda medir os tempos de resposta quando submetidos a um grande número de pedidos. Estes testes foram feitos recorrendo a aplicações para gerar notificações de eventos e aplicações, *listeners*, para as receber. Para gerar um grande número de notificações usaram-se *scripts* executados em *bash* e capturaram-se os pacotes na rede. Estas capturas de pacotes em conjunto com o comando *time* do *linux* permitiram a recolha da informação necessária para extrair os resultados.

A análise prática permitiu concluir/verificar diferenças significativas entre o SNMP, que usa o UDP na camada de transporte e transmite dados codificados em binário, e o WBEM e o NETCONF, que utilizam TCP e codificam os dados usando XML. O primeiro, por não necessitar de *tags* na sua descrição de dados, apresenta uma eficiência na sinalização muito superior aos segundos, que usam *tags* para evidenciar a hierarquia dos dados a transmitir. Consequentemente, o SNMP transmite substancialmente menos bytes, sensivelmente 40% menos que o NETCONF, o melhor dos protocolos que usam XML. Entre o WBEM e o NETCONF verifica-se que o segundo apresenta melhores resultados uma vez que o CIM-XML é extremamente verboso, mesmo quando comparado com o YIN, facto visível por comparação da **Caixa de Código 12** e da **Caixa de Código 13**. Em relação ao número de pacotes o SNMP também é o mais vantajoso, uma vez que no UDP não se confirma a receção de mensagens nem existe o *overhead* de criar ou terminar uma sessão. No NETCONF é criada uma sessão persistente por cada subscrição de eventos, impondo um *overhead* muito menor que no WBEM que cria uma sessão TCP por cada indicação de evento, o que penaliza bastante o seu desempenho. Quando necessitam de enviar um grande número de notificações de eventos, os valores dos tempos de resposta do WBEM aumentam bastante mais rápido que os dos restantes protocolos. O SNMP apresenta o melhor resultado mas o NETCONF apresenta uma degradação pouco significativa quando comparado com o SNMP.

O SNMP apresenta melhor desempenho nos testes, mas isso não suprime as desvantagens já anunciadas. O NETCONF apresenta capacidades que o WBEM não implementa, como a distinção entre dados de estado e configuração, suporte de vários repositórios de dados ou transações das configurações. Pela análise dos testes verificou-se que o YANG permite transmitir a mesma informação que no WBEM mas com uma verbosidade muito menor, o que confere ao NETCONF um desempenho muito superior ao do WBEM. As vantagens verificadas para as mensagens de notificação de eventos prevêem-se ainda mais notáveis com as restantes operações, onde a quantidade de informação a transmitir é maior. De acordo com [109], verifica-se que nestes casos o desempenho do SNMP chega a ser pior que o do WBEM, dando uma clara vantagem ao NETCONF.

No entanto, a sua adoção não depende exclusivamente do seu desempenho. O NETCONF ainda é um protocolo recente e com muito trabalho de normalização pela frente: por exemplo, ainda decorre a normalização de um modelo de eventos comuns a vários sistemas e do modelo de controlo de acessos, sendo este ultimo indispensável à adoção do NETCONF por parte dos administradores de sistemas. Posteriormente, o NETCONF pode vir a beneficiar de capacidades para descobrir dispositivos na rede, o que pode beneficiar a sua utilização, por exemplo em redes autonómicas.

A capacidade de notificação de eventos pode tirar partido de uma funcionalidade idêntica ao *hearbeat* do WS-MAN, para sinalizar subscrições de eventos ativas. A inclusão de uma operação *get-status* pode agilizar a sua utilização por parte dos administradores de sistemas.

Desde a sua génese, o NETCONF teve em atenção as necessidades existentes na gestão e monitorização de redes, sendo que a sua flexibilidade e capacidade de adaptação lhe conferem um grande potencial para ser implementado em redes heterogéneas, estando inclusivamente normalizada a sua aplicação em dispositivos com baixos recursos. É também uma mais-valia para a implementação de redes autonómicas, onde a facilidade em alterar configurações e a notificação de eventos podem prestar um grande apoio. Para a adoção do NETCONF é inevitável um esforço por parte das empresas em investir no

desenvolvimento de soluções com suporte para NETCONF, e em conjunto com as instituições de investigação e desenvolvimento providenciar os modelos de dados adequados e em tempo útil. Pela análise feita nesta dissertação pode-se concluir que o NETCONF é um sério candidato a substituir o WBEM nas soluções de gestão e monitorização.

Referências

[1] J. Kurose and K. Ross, *Computer Networks, A Top-Down Approach*, 4th ed.: Pearson Education, Inc., 2008.

[2] J. Yu and I. Ajarmeh, "An Empirical Study of the NETCONF Protocol," in *Sixth International Conference on Networking and Services*, 2010, pp. 253-258.

[3] "Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model," ISO, ISO/IEC 7498-1, 1994.

[4] (2011, July) ITU-T Home. [Online]. http://www.itu.int/ITU-T/

[5] The Internet Engineering Task Force (IETF). [Online]. http://www.ietf.org/

[6] (2011) Objectos Management Group. [Online]. http://www.omg.org/

[7] Internet Research Task Force (IRTF). [Online]. http://irtf.org/

[8] M. Fedor, M. Schoffstall and J. Davin J. Case, "A Simple Network Management Protocol (SNMP)," IETF, RFC 1157, May 1990.

[9] A. Pras and J. Martin-Flatin J. Schonwalder, "On the Future of Internet Management Technologies," *IEEE Communications Magazine*, pp. 90-97, October 2003.

[10] Internet Architecture Board. [Online]. http://www.iab.org/

[11] J. Schoenwaelder, "Overview of the 2002 IAB Network Management Workshop," IETF, Network WG, Informational RFC 3535, 05, 2003.

[12] Ed. D. Durham, "The COPS (Common Open Policy Service) Protocol," IETF, RFC 2748, Jan. 2000.

[13] DMTF, "Common Information Model (CIM) Infrastructure v2.6.0," DMTF, Specification DSP0004, 2010.

[14] R. Sprenkels and J. P. Martin-Flatin, "Bulk Transfers of MIB Data," École Polytechnique Fédéral de Lausanne, Lausanne, Technical Report SSC SSC/1999/009, 1999.

[15] K. McCloghrie and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets," IETF, Network WG, RFC 1156, 1990.

[16] (2011) Distributed Management Task Force, Inc. [Online]. http://www.dmtf.org/

[17] J. Martin-Flatin, *Web-Based Management of IP Networks and Systems*.: Wiley, 2002.

[18] H. Choi and J. Wong M. Choi, "XML-based Configuration Management for IP Network Devices," *IEEE Communications Magazine*, vol. 41, no. July, pp. 84-91, 2004.

[19] B. Zhang, G. Li, Y. Li and X. Gao Y. Gao, "The Comparison and Analisys of the Tree Data Model and Table-Like Data Model Based on NETCONF," in *2nd International Asia Conference on Informatics in Control, Automation and Robotics*, 2010, pp. 75-78.

[20] M. Bjorklund, P. Shafer J. Schonwalder, "Network Configuration Managment Using NETCONF and YANG," *IEEE Communications Magazine*, pp. 166-173, Sept. 2010.

[21] R. Enns, "NETCONF Configuration Protocol," IETF, RFC 4741, Dec. Dec. 2006.

[22] M. Wasserman, "Using the NETCONF Configuration Protocol over Secure SHell (SSH)," IETF, RFC 4742, Dec. 2006.

[23] E. Lear, "Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP)," IETF, RFC 4744, Dec. 2006.

[24] T. Goddard, "Using NETCONF over the Simple Object Access Protocol (SOAP)," IETF, RFC 4743, Dec. 2006.

[25] E. Rescorla T. Dierks, "The Transport Layer Security (TLS) Protocol, version 1.2," IETF, RFC 5246, 2008.

[26] S. Chisholm and H. Trevino, "NETCONF Event Notifications," IETF, RFC 5277, 2008.

[27] B. Lengyel and M. Bjorklund, "Partial Lock Remote Procedure Call (RPC) for NETCONF," IETF, RFC 5717, 2009.

[28] M. Scott and M. Bjorklund, "YANG Module for NETCONF Monitoring," IETF, RFC 6022, 2010.

[29] M. Bjorklund, , J. Schoenwaelder and A. Bierman R. Enns, "Network Configuration Protocol

(NETCONF)," IETF, RFC 6241, 2011.

[30] M. Wasserman, "Using the NETCONF Protocol over Secure Shell (SSH)," IETF, RFC 6242, 2011.

[31] A. Bierman and B. Lengyel, "With-defaults Capability for NETCONF," IETF, RFC 6243, 2011.

[32] D. Perkins and J. Schoenwaelder K. McCloghrie, "Structure of Management Information Version 2 (SMIv2)," IETF, RFC 2578, April 1999.

[33] ITU-T. (2011, March) Abstract Syntax Notation One (ASN.1). [Online]. http://www.itu.int/ITU-T/studygroups/com17/languages/

[34] and K. McCloghrie M. Rose, "Structure and Identification of Management Information for TCP/IP-based Internets," IETF, RFC 1155, May 1990.

[35] K. McCloghrie, , M. Rose and S. Waldbusser J. Case, "Introduction to version 2 of the Internet-standard Network Management Framework," IETF, RFC 1441, 1993.

[36] Douglas Mauro and Kevin Schmidt, *Essential SNMP*.: O'Reilly Media, 2001.

[37] K. McCloghrie and J. Galvin, "Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)," IETF, RFC 1447, 1993.

[38] R. Presuhn and B. Wijnen D. Harrington, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," IETF, standard RFC 2571, 2002.

[39] M. Bjorklund and P. Shafer J. Schonwalder, "Network Configuration Management Using NETCONF and YANG," *IEEE Communications Magazine*, pp. 166-173, September 2010.

[40] J. Schoenwaelder, "Simple Network Management Protocol (SNMP) over Transmission Control Protocol (TCP) Transport Mapping," IETF, Experimental RFC 3430, 2002.

[41] D. Xiao, H. Xu Y. Chang, "Design and Implementation of NETCONF-Based Network Management System," in *Second International Conference on Future Generation Communication and Networking*, 2008, pp. 256-259.

[42] A. Hamid, T. Song and K. Asami Y. Kawahara, "Network Management Architecture Toward Universal Communication," in *3rd Internacional Universal Communication Symposium*, 2009, pp. 172-175.

[43] (2011) Distributed Management Task Force, Inc. [Online]. http://www.dmtf.org/

[44] DMTF. (2011) Web-Based Enterprise Management. [Online]. http://www.dmtf.org/standards/wbem

[45] DMTF. (2011) Web Services Management. [Online]. http://www.dmtf.org/standards/wsman

[46] DMTF. (2011) DMTF: Standards and Technology. [Online]. http://www.dmtf.org/standards

[47] DMTF. (2011) Common Information Model. [Online]. http://www.dmtf.org/standards/cim

[48] DMTF, "Representation of CIM in XML, v2.3.1," DMTF, Standard DSP0201, 2009.

[49] J. Gettys, J. Mogul R. Fielding, "Hypertext Transfer Protocol -- HTTP/1.1," IETF, RFC 2616, 1999.

[50] DMTF, "UML Profile for CIM," DMTF, Standard DSP0219, 2009.

[51] M. Choi, S. Yoo, J. Hong, H.Cho, C. Ahn and S. Jung So. Lee, "Design of a WBEM-based Management System for Ubiquitous Computing Servers,".

[52] Hewlett-Packard Development Company, L.P. (2011) HP WBEM Solutions. [Online]. http://h71028.www7.hp.com/enterprise/cache/9920-0-0-225-121.html

[53] ORACLE. (2010) About Solaris WBEM Services. [Online]. http://download.oracle.com/docs/cd/E19963-01/html/821-1605/chap-ov-8.html

[54] Cisco. (2011) Cisco MDS 9000 NX-OS Software Release 5.2(1). [Online]. http://www.cisco.com/en/US/prod/collateral/ps4159/ps6409/ps5989/ps11592/data_sheet_c78-639741.html

[55] Microsoft. Windows Management Instrumentation. [Online]. http://msdn.microsoft.com/en-us/library/aa394582(v=vs.85).aspx

[56] Microsoft. (2011) WMI Providers. [Online]. http://msdn.microsoft.com/en-us/library/aa394570(v=vs.85).aspx

[57] WBEM Solutions Inc. (2010) WBEM Solutions. [Online]. http://www.wbemsolutions.com/

[58] IBM Corporation. IBM Director tasks. [Online]. http://publib.boulder.ibm.com/infocenter/director/v5r2/index.jsp?topic=/diricinfo_5.20/fqm0_r_ibm_

director_tasks.html

[59] DELL. (2011) Common Information Model (CIM) Solution. [Online]. http://support.dell.com/support/edocs/storage/Non_RAID/p46371/cim.htm#cimb

[60] JSR48. (2011, June) WBEM Services. [Online]. http://sourceforge.net/projects/wbemservices/

[61] The Open Group. (2011) OpenPegasus "C++ CIM/WBEM Manageability Services Broker". [Online]. http://www.openpegasus.org/

[62] OpenWBEM project. OPENWBEM. [Online]. http://openwbem.sourceforge.net

[63] (2009) SBLIM. [Online]. http://sourceforge.net/apps/mediawiki/sblim/index.php?title=Main_Page

[64] Michael Chase-Salerno and Narasimha Sharoff Chris Buccella. (2011) Small Footprint CIM Broker (SFCB). [Online]. http://sourceforge.net/apps/mediawiki/sblim/index.php?title=Sfcb

[65] KDE CIM Browser project. (2004) KDE CIM Browser. [Online]. http://kde-cim.sourceforge.net/

[66] DuskFire, Inc. (2007) CIM Navigator. [Online]. http://cimnavigator.com/

[67] OpenPegasus Project. (2003) pegasus-JavaCIMClient. [Online]. http://cvs.opengroup.org/cgi-bin/viewcvs.cgi/pegasus-JavaCIMClient/

[68] (2009, July) Yet Another WBEM Navigator (YAWN). [Online]. http://sourceforge.net/apps/mediawiki/pywbem/index.php?title=YAWN

[69] Waiki Wright. (2009, October) ECUTE. [Online]. http://sourceforge.net/apps/mediawiki/sblim/index.php?title=Ecute

[70] The WBEMsource initiative. [Online]. http://www.wbemsource.org/

[71] Purgos / SofTulz.Net. [Online]. http://www.softulz.net/

[72] W3C. (2011) W3C. [Online]. http://www.w3.org/

[73] C. Liu D. Both, "Web Services Description Language (WSDL) Version 2.0," W3C, W3C Recomendation 2007.

[74] G. Silvestrin, R. Sanchez, L. Gaspary and L. Granville G. Moura, "On the Performance of Web Services Management Standards - An Evaluation of MUWS and WS-Management for Network Management," Federal University of Rio Grande do Sul, Porto Alegre, Brazil, 2007.

[75] OASIS®. (2011) OASIS. [Online]. http://www.oasis-open.org/

[76] OASIS, "Web Services Distributed Management: Management Using Web Services (MUWS 1.0) Part 1," OASIS, OASIS Standard wsdm-muws-part1-1.0, 2005.

[77] OASIS, "Web Services Distributed Management: Management Using Web Services (MUWS 1.0) Part 2," OASIS, OASIS standard wsdm-muws-part2-1.0, 2005.

[78] WBEM Infrastructure & Protocols Working Group, "Web Services for Management (WS-Management) Specification," DMTF, Specification DSP0226, March 2010.

[79] OASIS Web Services Distributed Management TC, "Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.1," OASIS, OASIS Standard wsdm-mows-1.1-spec-os-01, 2006.

[80] OASIS, "An Introduction to WSDM," OASIS, Committee Draft wsdm-1.0-intro-primer-cd-01, 2006.

[81] W3C. (2004) SOAP Specifications. [Online]. http://www.w3.org/TR/soap/

[82] W3C. (2010) [Online]. http://www.w3.org/standards/xml/schema

[83] W3C. (2006, May) Web Services Addressing 1.0 – Core. [Online]. http://www.w3.org/TR/ws-addr-core/

[84] OASIS, "Web Services Resource 1.2 (WS-Resource) ," OASIS, OASIS standard wsrf-ws_resource-1.2-spec-os, 2006.

[85] OASIS, "Web Services Resource Properties 1.2 (WS-ResourceProperties)," OASIS, OASIS standard wsrf-ws_resource_properties-1.2-spec-os, 2006.

[86] OASIS, "Web Services Resource Lifetime 1.2 (WS-ResourceLifetime) ," OASIS, OASIS standard wsrf-ws_resource_lifetime-1.2-spec-os, 2006.

[87] OASIS, "Web Services Service Group 1.2 (WS-ServiceGroup)," OASIS, OASIS standard wsrf-ws_service_group-1.2-spec-os, 2006.

[88] OASIS, "Web Services Base Faults 1.2 (WS-BaseFaults)," OASIS, OASIS standard wsrf-

ws_base_faults-1.2-spec-os, 2006.

[89] OASIS, "Web Services Base Notification 1.3 (WS-BaseNotification)," OASIS, OASIS standard wsn-ws_base_notification-1.3-spec-os, 2006.

[90] OASIS, "Web Services Brokered Notification 1.3 (WS-BrokeredNotification)," OASIS, OASIS standard wsn-ws_brokered_notification-1.3-spec-os, 2006.

[91] OASIS, "Web Services Topics 1.3 (WS-Topics)," OASIS, OASIS standard wsn-ws_topics-1.3-spec-os, 2006.

[92] OASIS, "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)," OASIS, Oasis standard specification wss-v1.1-spec-os-SOAPMessageSecurity, 2006.

[93] J. and S. DeRose Clark, "XML Path Language (XPath) Version 1.0," World Wide Web Consortium, http://www.w3.org/TR/1999/REC-xpath-19991116, 1999.

[94] "Information technology -- Database languages -- SQL -- Part 1: Framework (SQL/Framework)," ISO, ISO/IEC 9075-1:2008, 2008.

[95] DMTF, "CIM Query Language Specification v1.0," Specification DSP0202, 2007.

[96] W3C. (2006, March) Web Services Eventing (WS-Eventing). [Online]. http://www.w3.org/Submission/WS-Eventing/

[97] DELL. (2009) Integrated Dell™ Remote Access Controller 6 (iDRAC6) Version 1.0 User Guide. [Online]. http://support.dell.com/support/edocs/software/smdrac3/idrac/idrac10mono/en/ug/html/index.htm

[98] DELL. (2011) Using the WS-MAN Interface. [Online]. http://support.dell.com/support/edocs/software/smdrac3/idrac/idrac10mono/en/ug/html/racugc1d.htm

[99] Microsoft. (2011) Windows Remote Management. [Online]. http://msdn.microsoft.com/en-us/library/aa384426(v=VS.85).aspx

[100] Intel. (2009) Intel® Active Management Technology (AMT). [Online]. http://software.intel.com/en-us/articles/intel-active-management-technology/

[101] Intel Corporation. (2010) Intel® WS-Management Translator. [Online]. http://software.intel.com/en-us/articles/intel-ws-management-translator/

[102] Intel Corporation. (2008) WS-Management and Intel® Active Management Technology: A Primer. [Online]. http://software.intel.com/en-us/articles/ws-management-and-intel-active-management-technology-a-primer/

[103] Wash., and WALTHAM, Mass. REDMOND. (2007) Microsoft and Novell Announce Technical Collaboration for Customers. [Online]. http://www.novell.com/news/press/2007/2/microsoft_and_novell_announce_technical_collaboration_for_customers.html

[104] kkaempf, nashif and tariqx jerry_yu. (2011) open wsman. [Online]. http://sourceforge.net/projects/openwsman/

[105] (2011) A Java implementation of WS-Management. [Online]. http://java.net/projects/wiseman/

[106] OpenPegasus. (2008) Pegasus Enhancement Proposal (PEP). [Online]. http://www.openpegasus.org/pp/uploads/40/16744/PEP311_WSManSupportInCIMServer_1_4.htm

[107] T. Drevers, R. Meent and D. Quartel A. Pras, "Comparing the Performance of SNMP and Web Services-based Management," in *eTransactions on Network and service Management*, 2004.

[108] R. L. Vianna, L. Granville, M. Almeida and L. Tarouco R. Neisse, "IMplementation and Bandwidth Comsuption Evaluation of SNMP to Web Services Gateways," in *9th IEEE/IFIP Network Operations and Management Symposium*, South Korea, 2004, pp. 715-728.

[109] J. Oliveira, R. Aguiar P. Gonçalves, "An evaluation of network managment protocols," in *IFIP/IEEE International Symposium on Integrated Network Management*, 2009, pp. 537-544.

[110] M. Badra, "NETCONF over Transport Layer Security (TLS)," IETF, RFC 5539, May 2009.

[111] Ed. M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," IETF, RFC 6020, 2010.

[112] NETMOD Working Group. (2011) NETCONF Data Modeling Language (netmod). [Online]. http://datatracker.ietf.org/wg/netmod/charter/

[113] Hui and Xiao, Debao Xu, "Data Modeling for NETCONF-Based Network Managment: XML or Yang," in *11th IEEE International Conference on Communication Technology Proceedings*, Wuhan, P.R. China, 2008, pp. 561-564.

[114] P. Shafer, "An Architecture for Network Management Using NETCONF and YANG," IETF, Informational RFC 6244, 2001.

[115] Ed. J. Schoenwaelder, "Common YANG Data Types," IEEE, RFC 6021, 2010.

[116] A. Bierman and B. Lengyel, "With-defaults Capability for NETCONF," IETF, standard RFC 6243, 2011.

[117] Tail-f Systems. (2011) ConfD NETCONF Agent Datasheet. [Online]. http://www.tail-f.com/confd-netconf-datasheet

[118] Andy Bierman. (2011) netconf central. [Online]. http://www.netconfcentral.org/yuma

[119] Giuseppe Palmeri. (2010) netconf4android. [Online]. http://code.google.com/p/netconf4android/

[120] CESNET z.s.p.o. (2011) netopeer. [Online]. http://code.google.com/p/netopeer/

[121] CESNET, z.s.p.o. (2009) FlowMon probe. [Online]. http://www.liberouter.org/flowmon/index.php

[122] CESNET, z.s.p.o. (2009) Flexible FlowMon probe. [Online]. http://www.liberouter.org/fflowmon/index.php

[123] R. State. (2009, July) YENCA - A XML based network management framework compatible with the IETF NetConf specification. [Online]. http://sourceforge.net/projects/yenca/

[124] LORIA-INRIA Lorraine. (2011, February) EnSuite, a Netconf Framework. [Online]. http://ensuite.sourceforge.net/

[125] Shikhar Bhushan. (2009) ncclient - Python library for NETCONF clients. [Online]. ncclient

[126] Tail-f Systems. (2011) Tail-f - Products & Services Overview. [Online]. http://www.tail-f.com/products-and-services/overview

[127] GoAhead Software Inc. (2011) GoAhead embeddedMIND. [Online]. http://goahead.com/products/embeddedmind/embedded-solutions.aspx

[128] Cisco Systems, Inc. (2006) NETCONF over SSHv2. [Online]. http://www.cisco.com/en/US/docs/ios/12_2sr/12_2sra/feature/guide/srnetcon.html

[129] Cisco Systems, Inc. (2006) NETCONF over BEEP. [Online]. http://www.cisco.com/en/US/docs/ios/12_4t/12_4t11/htnetbe.html#wp1049404

[130] Cisco Systems, Inc. NETCONF Client GUI. [Online]. http://www.cisco.com/en/US/docs/net_mgmt/enhanced_device_interface/2.2/developer/guide/ntcfapp.html

[131] Juniper Networks, Inc. (2011) NETCONF XML Management Protocol. [Online]. http://www.juniper.net/support/products/netconf/

[132] B. Zhang, G. Li, Y. Li, X. Gao Z. Yang, "The Implementation and Comparison Analysis of Subtree Filtering and XPath Capability in NETCONF," in *IC-NIDC*, 2009, pp. 921-925.

[133] A. Bierman and M. Bjorklund, "Network Configuration Protocol Access Control Model," IETF, Internet-Draft draft-ietf-netconf-access-control-04, 2011.

[134] F. Muller and S. Fisher T. Klie, "Network Monitoring with Asynchronous Notifications in Web Service Environments," Tech. Uni. of Braunshweig, Uni. Dusseldorf and Uni. zu Lubeck, 2007.

[135] B. Zhang, G. Li, Y. Li and X. Gao R. Wang, "The Implementation and Analisis of the Monitoring Module based on NETCONF," in *International Symposium on Information and Engineering*, 2008, pp. 12-15.

[136] Robert Van Engelen, "Code Generation Techniques for Developing Light-Weight XML Web Services for Embedded Devices," in *ACM SIGAPP SAC Conference (Embedded Systems Track)*, Nicosia, Cyprus, 2004.

[137] M. Aiello and S. Dustdar D. Schall, "Web Services on Embedded Devices," *J. WEB INFOR. SYST.*, vol. 1, no. 1, March 2005.

[138] Robert A. van Engelen. (2011, May) The gSOAP Toolkit for SOAP Web Services and XML-Based Applications. [Online]. http://www.cs.fsu.edu/~engelen/soap.html

[139] Robert A. van Engelen and Kyle Gallivan, "The gSOAP Toolkit for Web Services and Peer-To-Peer

Computing Networks," in *2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002)*, Berlin, Germany, 2002, pp. 128-135.

[140] V. Perelman and J. Schoenwaelder, "Network Configuration Protocol for Constrained Devices (NETCONF Light)," IETF, Internet-Draft 2011.

[141] Network Group at CMU. (2011, May) NET-SNMP. [Online]. http://www.net-snmp.org

[142] W. Wang, K. Peters, D. Gelman and J. Dimarogonas K. Weinstein, "A Domain-Level Data Model for Automating Network Configuration," in *The 2010 Military Communications Conference, Cyber Security and Network Management*, 2010, pp. 1337-1342.

Anexos

A. Netconf.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:tns="urn:ietf:params:xml:ns:netconf:soap:1.0"
        xmlns:netb="urn:ietf:params:xml:ns:netconf:base:1.0"
        targetNamespace="urn:ietf:params:xml:ns:netconf:soap:1.0"
        name="netconf-soap_1.0.wsdl">

        <import namespace="urn:ietf:params:xml:ns:netconf:base:1.0"
                location="./netconf.xsd" />

        <message name="helloRequest">
                <part name="in" element="netb:hello"/>
        </message>
        <message name="helloResponse">
                <part name="out" element="netb:hello"/>
        </message>
        <message name="rpcRequest">
                <part name="in" element="netb:rpc"/>
        </message>
        <message name="rpcResponse">
                <part name="out" element="netb:rpc-reply"/>
        </message>

        <portType name="netconfPortType">
                <operation name="rpc">
                        <input message="tns:rpcRequest"/>
                        <output message="tns:rpcResponse"/>
                </operation>
                <operation name="hello">
                        <input message="tns:helloRequest"/>
                        <output message="tns:helloResponse"/>
                </operation>
        </portType>

        <binding name="netconfBinding" type="tns:netconfPortType">
                <SOAP:binding style="document"
                        transport="http://schemas.xmlsoap.org/soap/http"/>
                <operation name="hello">
                        <SOAP:operation/>
                        <input>
                                <SOAP:body use="literal"
                                        namespace="urn:ietf:params:xml:ns:netconf:soap:1.0"/>
                        </input>
                        <output>
                                <SOAP:body use="literal"
                                        namespace="urn:ietf:params:xml:ns:netconf:soap:1.0"/>
                        </output>
```

```
                </operation>
                <operation name="rpc">
                        <SOAP:operation/>
                        <input>
                                <SOAP:body use="literal"
                                        namespace="urn:ietf:params:xml:ns:netconf:base:1.0"/>
                        </input>
                        <output>
                                <SOAP:body use="literal"
                                        namespace="urn:ietf:params:xml:ns:netconf:base:1.0"/>
                        </output>
                </operation>
        </binding>
</definitions>
```

B. Netconf.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
        targetNamespace="urn:ietf:params:xml:ns:netconf:base:1.0"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified"
        xml:lang="en">
        <!--
                import standard XML definitions
        -->
        <xs:import namespace="http://www.w3.org/XML/1998/namespace"
                schemaLocation="http://www.w3.org/2001/xml.xsd">
                <xs:annotation>
                        <xs:documentation>
                                This import accesses the xml: attribute groups for the
                                xml:lang as declared on the error-message element.
                        </xs:documentation>
                </xs:annotation>
        </xs:import>
        <!--
                message-id attribute
        -->
        <xs:simpleType name="messageIdType">
                <xs:restriction base="xs:string">
                        <xs:maxLength value="4095"/>
                </xs:restriction>
        </xs:simpleType>
        <!--
                Types used for session-id
        -->
        <xs:simpleType name="SessionId">
                <xs:restriction base="xs:unsignedInt">
                        <xs:minInclusive value="1"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="SessionIdOrZero">
                <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
        <!--
                <rpc> element
        -->
        <xs:complexType name="rpcType">
                <xs:sequence>
                        <xs:element ref="rpcOperation"/>
                </xs:sequence>
                <xs:attribute name="message-id" type="messageIdType"
                        use="required"/>
                <!--
                        Arbitrary attributes can be supplied with <rpc> element.
                -->
                <xs:anyAttribute processContents="lax"/>
        </xs:complexType>
        <xs:element name="rpc" type="rpcType"/>
        <!--
```

```
                data types and elements used to construct rpc-errors
        -->
        <xs:simpleType name="ErrorType">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="transport"/>
                        <xs:enumeration value="rpc"/>
                        <xs:enumeration value="protocol"/>
                        <xs:enumeration value="application"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="ErrorTag">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="in-use"/>
                        <xs:enumeration value="invalid-value"/>
                        <xs:enumeration value="too-big"/>
                        <xs:enumeration value="missing-attribute"/>
                        <xs:enumeration value="bad-attribute"/>
                        <xs:enumeration value="unknown-attribute"/>
                        <xs:enumeration value="missing-element"/>
                        <xs:enumeration value="bad-element"/>
                        <xs:enumeration value="unknown-element"/>
                        <xs:enumeration value="unknown-namespace"/>
                        <xs:enumeration value="access-denied"/>
                        <xs:enumeration value="lock-denied"/>
                        <xs:enumeration value="resource-denied"/>
                        <xs:enumeration value="rollback-failed"/>
                        <xs:enumeration value="data-exists"/>
                        <xs:enumeration value="data-missing"/>
                        <xs:enumeration value="operation-not-supported"/>
                        <xs:enumeration value="operation-failed"/>
                        <xs:enumeration value="partial-operation"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="ErrorSeverity">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="error"/>
                        <xs:enumeration value="warning"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:complexType name="errorInfoType">
                <xs:sequence>
                        <xs:choice>
                                <xs:element name="session-id" type="SessionIdOrZero"/>
                                <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                        <xs:sequence>
                                                <xs:element name="bad-attribute" type="xs:QName"
                                                        minOccurs="0" maxOccurs="1"/>
                                                <xs:element name="bad-element" type="xs:QName"
                                                        minOccurs="0" maxOccurs="1"/>
                                                <xs:element name="ok-element" type="xs:QName"
                                                        minOccurs="0" maxOccurs="1"/>
                                                <xs:element name="err-element" type="xs:QName"
                                                        minOccurs="0" maxOccurs="1"/>
                                                <xs:element name="noop-element" type="xs:QName"
                                                        minOccurs="0" maxOccurs="1"/>
                                                <xs:element name="bad-namespace"
type="xs:QName"
                                                        minOccurs="0" maxOccurs="1"/>
```

```
                                        </xs:sequence>
                                </xs:sequence>
                        </xs:choice>
                        <!-- elements from any other namespace are also allowed
                                to follow the NETCONF elements -->
                        <xs:any namespace="##other"
                                minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="rpcErrorType">
                <xs:sequence>
                        <xs:element name="error-type" type="ErrorType"/>
                        <xs:element name="error-tag" type="ErrorTag"/>
                        <xs:element name="error-severity" type="ErrorSeverity"/>
                        <xs:element name="error-app-tag" type="xs:string"
                                minOccurs="0"/>
                        <xs:element name="error-path" type="xs:string" minOccurs="0"/>
                        <xs:element name="error-message" minOccurs="0">
                                <xs:complexType>
                                        <xs:simpleContent>
                                                <xs:extension base="xs:string">
                                                        <xs:attribute ref="xml:lang" use="optional"/>
                                                </xs:extension>
                                        </xs:simpleContent>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="error-info" type="errorInfoType"
                                minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
        <!--
                <rpc-reply> element
        -->
        <xs:complexType name="rpcReplyType">
                <xs:choice>
                        <xs:element name="ok"/>
                        <xs:group ref="rpcResponse"/>
                </xs:choice>
                <xs:attribute name="message-id" type="messageIdType"
                        use="optional"/>
                <!--
                        Any attributes supplied with <rpc> element must be returned
                        on <rpc-reply>.
                -->
                <xs:anyAttribute processContents="lax"/>
        </xs:complexType>
        <xs:group name="rpcResponse">
                <xs:sequence>
                        <xs:element name="rpc-error" type="rpcErrorType"
                                minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="data" type="dataInlineType" minOccurs="0"/>
                </xs:sequence>
        </xs:group>
        <xs:element name="rpc-reply" type="rpcReplyType"/>
        <!--
                Type for <test-option> parameter to <edit-config>
        -->
        <xs:simpleType name="testOptionType">
```

```
        <xs:restriction base="xs:string">
            <xs:enumeration value="test-then-set"/>
            <xs:enumeration value="set"/>
        </xs:restriction>
</xs:simpleType>
<!--
        Type for <error-option> parameter to <edit-config>
-->
<xs:simpleType name="errorOptionType">
        <xs:restriction base="xs:string">
            <xs:annotation>
                <xs:documentation>
                    Use of the rollback-on-error value requires
                    the :rollback-on-error capability.
                </xs:documentation>
            </xs:annotation>
            <xs:enumeration value="stop-on-error"/>
            <xs:enumeration value="continue-on-error"/>
            <xs:enumeration value="rollback-on-error"/>
        </xs:restriction>
</xs:simpleType>
<!--
        rpcOperationType: used as a base type for all
        NETCONF operations
-->
<xs:complexType name="rpcOperationType"/>
<xs:element name="rpcOperation"
        type="rpcOperationType" abstract="true"/>
<!--
        Type for <config> element
-->
<xs:complexType name="configInlineType">
        <xs:complexContent>
            <xs:extension base="xs:anyType"/>
        </xs:complexContent>
</xs:complexType>
<!--
        Type for <data> element
-->
<xs:complexType name="dataInlineType">
        <xs:complexContent>
            <xs:extension base="xs:anyType"/>
        </xs:complexContent>
</xs:complexType>
<!--
        Type for <filter> element
-->
<xs:simpleType name="FilterType">
        <xs:restriction base="xs:string">
            <xs:annotation>
                <xs:documentation>
                    Use of the xpath value requires the :xpath capability.
                </xs:documentation>
            </xs:annotation>
            <xs:enumeration value="subtree"/>
            <xs:enumeration value="xpath"/>
        </xs:restriction>
</xs:simpleType>
```

```
<xs:complexType name="filterInlineType">
        <xs:complexContent>
                <xs:extension base="xs:anyType">
                        <xs:attribute name="type"
                                type="FilterType" default="subtree"/>
                        <!-- if type="xpath", the xpath expression
                                appears in the select element -->
                        <xs:attribute name="Text"/>  <!--AQUI-->
                        <xs:attribute name="select"/>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
<!--
        configuration datastore names
-->
<xs:annotation>
        <xs:documentation>
                The startup datastore can be used only if the :startup
                capability is advertised. The candidate datastore can
                be used only if the :candidate datastore is advertised.
        </xs:documentation>
</xs:annotation>
<xs:complexType name="configNameType"/>
<xs:element name="config-name"
        type="configNameType" abstract="true"/>
<xs:element name="startup" type="configNameType"
        substitutionGroup="config-name"/>
<xs:element name="candidate" type="configNameType"
        substitutionGroup="config-name"/>
<xs:element name="running" type="configNameType"
        substitutionGroup="config-name"/>
<!--
        operation attribute used in <edit-config>
-->
<xs:simpleType name="editOperationType">
        <xs:restriction base="xs:string">
                <xs:enumeration value="merge"/>
                <xs:enumeration value="replace"/>
                <xs:enumeration value="create"/>
                <xs:enumeration value="delete"/>
        </xs:restriction>
</xs:simpleType>
<xs:attribute name="operation"
        type="editOperationType" default="merge"/>
<!--
        <default-operation> element
-->
<xs:simpleType name="defaultOperationType">
        <xs:restriction base="xs:string">
                <xs:enumeration value="merge"/>
                <xs:enumeration value="replace"/>
                <xs:enumeration value="none"/>
        </xs:restriction>
</xs:simpleType>
<!--
        <url> element
-->
<xs:complexType name="configURIType">
```

```
            <xs:annotation>
                <xs:documentation>
                    Use of the url element requires the :url capability.
                </xs:documentation>
            </xs:annotation>
            <xs:simpleContent>
                <xs:extension base="xs:anyURI"/>
            </xs:simpleContent>
</xs:complexType>
<!--
            Type for <source> element (except <get-config>)
-->
<xs:complexType name="rpcOperationSourceType">
            <xs:choice>
                <xs:element name="config" type="configInlineType"/>
                <xs:element ref="config-name"/>
                <xs:element name="url" type="configURIType"/>
            </xs:choice>
</xs:complexType>
<!--
            Type for <source> element in <get-config>
-->
<xs:complexType name="getConfigSourceType">
            <xs:choice>
                <xs:element ref="config-name"/>
                <xs:element name="url" type="configURIType"/>
            </xs:choice>
</xs:complexType>
<!--
            Type for <target> element
-->
<xs:complexType name="rpcOperationTargetType">
            <xs:choice>
                <xs:element ref="config-name"/>
                <xs:element name="url" type="configURIType"/>
            </xs:choice>
</xs:complexType>
<!--
            <get-config> operation
-->
<xs:complexType name="getConfigType">
            <xs:complexContent>
                <xs:extension base="rpcOperationType">
                    <xs:sequence>
                        <xs:element name="source"
                            type="getConfigSourceType"/>
                        <xs:element name="filter"
                            type="filterInlineType" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
</xs:complexType>
<xs:element name="get-config" type="getConfigType"
            substitutionGroup="rpcOperation"/>
<!--
            <edit-config> operation
-->
<xs:complexType name="editConfigType">
```

```
        <xs:complexContent>
                <xs:extension base="rpcOperationType">
                        <xs:sequence>
                                <xs:annotation>
                                        <xs:documentation>
                                                Use of the test-option element requires the
                                                :validate capability. Use of the url element
                                                requires the :url capability.
                                        </xs:documentation>
                                </xs:annotation>
                                <xs:element name="target"
                                        type="rpcOperationTargetType"/>
                                <xs:element name="default-operation"
                                        type="defaultOperationType"
                                        minOccurs="0"/>
                                <xs:element name="test-option"
                                        type="testOptionType"
                                        minOccurs="0"/>
                                <xs:element name="error-option"
                                        type="errorOptionType"
                                        minOccurs="0"/>
                                <xs:choice>
                                        <xs:element name="config"
                                                type="configInlineType"/>
                                        <xs:element name="url"
                                                type="configURIType"/>
                                </xs:choice>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
<xs:element name="edit-config" type="editConfigType"
        substitutionGroup="rpcOperation"/>
<!--
        <copy-config> operation
-->
<xs:complexType name="copyConfigType">
        <xs:complexContent>
                <xs:extension base="rpcOperationType">
                        <xs:sequence>
                                <xs:element name="target" type="rpcOperationTargetType"/>
                                <xs:element name="source" type="rpcOperationSourceType"/>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
<xs:element name="copy-config" type="copyConfigType"
        substitutionGroup="rpcOperation"/>
<!--
        <delete-config> operation
-->
<xs:complexType name="deleteConfigType">
        <xs:complexContent>
                <xs:extension base="rpcOperationType">
                        <xs:sequence>
                                <xs:element name="target" type="rpcOperationTargetType"/>
                        </xs:sequence>
                </xs:extension>
```

```
        </xs:complexContent>
</xs:complexType>
<xs:element name="delete-config" type="deleteConfigType"
        substitutionGroup="rpcOperation"/>
<!--
        <get> operation
-->
<xs:complexType name="getType">
        <xs:complexContent>
                <xs:extension base="rpcOperationType">
                        <xs:sequence>
                                <xs:element name="filter"
                                        type="filterInlineType" minOccurs="0"/>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
<xs:element name="get" type="getType"
        substitutionGroup="rpcOperation"/>
<!--
        <lock> operation
-->
<xs:complexType name="lockType">
        <xs:complexContent>
                <xs:extension base="rpcOperationType">
                        <xs:sequence>
                                <xs:element name="target"
                                        type="rpcOperationTargetType"/>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
<xs:element name="lock" type="lockType"
        substitutionGroup="rpcOperation"/>
<!--
        <unlock> operation
-->
<xs:complexType name="unlockType">
        <xs:complexContent>
                <xs:extension base="rpcOperationType">
                        <xs:sequence>
                                <xs:element name="target" type="rpcOperationTargetType"/>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
<xs:element name="unlock" type="unlockType"
        substitutionGroup="rpcOperation"/>
<!--
        <validate> operation
-->
<xs:complexType name="validateType">
        <xs:annotation>
                <xs:documentation>
                        The validate operation requires the :validate capability.
                </xs:documentation>
        </xs:annotation>
        <xs:complexContent>
```

```
                    <xs:extension base="rpcOperationType">
                        <xs:sequence>
                            <xs:element name="source" type="rpcOperationSourceType"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:element name="validate" type="validateType"
                substitutionGroup="rpcOperation"/>
            <!--
                <commit> operation
            -->
            <xs:simpleType name="confirmTimeoutType">
                <xs:restriction base="xs:unsignedInt">
                    <xs:minInclusive value="1"/>
                </xs:restriction>
            </xs:simpleType>
            <xs:complexType name="commitType">
                <xs:annotation>
                    <xs:documentation>
                        The commit operation requires the :candidate capability.
                    </xs:documentation>
                </xs:annotation>
                <xs:complexContent>
                    <xs:extension base="rpcOperationType">
                        <xs:sequence>
                            <xs:annotation>
                                <xs:documentation>
                                    Use of the confirmed and confirm-timeout
elements
                                    requires the :confirmed-commit capability.
                                </xs:documentation>
                            </xs:annotation>
                            <xs:element name="confirmed" minOccurs="0"/>
                            <xs:element name="confirm-timeout"
                                type="confirmTimeoutType"
                                minOccurs="0"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:element name="commit" type="commitType"
                substitutionGroup="rpcOperation"/>
            <!--
                <discard-changes> operation
            -->
            <xs:complexType name="discardChangesType">
                <xs:annotation>
                    <xs:documentation>
                        The discard-changes operation requires the
                        :candidate capability.
                    </xs:documentation>
                </xs:annotation>
                <xs:complexContent>
                    <xs:extension base="rpcOperationType"/>
                </xs:complexContent>
            </xs:complexType>
            <xs:element name="discard-changes"
```

```
                type="discardChangesType"
                substitutionGroup="rpcOperation"/>
        <!--
                <close-session> operation
        -->
        <xs:complexType name="closeSessionType">
                <xs:complexContent>
                        <xs:extension base="rpcOperationType"/>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="close-session" type="closeSessionType"
                substitutionGroup="rpcOperation"/>
        <!--
                <kill-session> operation
        -->
        <xs:complexType name="killSessionType">
                <xs:complexContent>
                        <xs:extension base="rpcOperationType">
                                <xs:sequence>
                                        <xs:element name="session-id"
                                                type="SessionId" minOccurs="1"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="kill-session" type="killSessionType"
                substitutionGroup="rpcOperation"/>
        <!--
                <hello> element
        -->
        <xs:element name="hello">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="capabilities">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="capability"
type="xs:anyURI"
                                                                maxOccurs="unbounded"/>
                                                </xs:sequence>
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="session-id"
                                        type="SessionId" minOccurs="0"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <!-- <create-subscription> operation -->
        <xs:complexType name="createSubscriptionType">
                <xs:complexContent>
                        <xs:extension base="rpcOperationType">
                                <xs:sequence>
                                        <xs:element name="stream"
                                                type="streamNameType" minOccurs="0">
                                                <xs:annotation>
                                                        <xs:documentation>
                                                                An optional parameter that indicates
                                                                which stream of events is of interest.
```

```
                                                    If not present, then events in the
                                                    default NETCONF stream will be
sent.
                                                </xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                        <xs:element name="filter"
                                            type="filterInlineType"
                                            minOccurs="0">
                                            <xs:annotation>
                                                <xs:documentation>
                                                    An optional parameter that indicates
                                                    which subset of all possible events
                                                    is of interest. The format of this
                                                    parameter is the same as that of the
                                                    filter parameter in the NETCONF
                                                    protocol operations. If not
                                                    present, all events not precluded
                                                    by other parameters will be sent.
                                                </xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                        <xs:element name="startTime" type="xs:dateTime"
                                            minOccurs="0" >
                                            <xs:annotation>
                                                <xs:documentation>
                                                    A parameter used to trigger the
replay
                                                    feature indicating that the replay
                                                    should start at the time specified. If
                                                    start time is not present, this is not a
                                                    replay subscription.
                                                </xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                        <xs:element name="stopTime" type="xs:dateTime"
                                            minOccurs="0" >
                                            <xs:annotation>
                                                <xs:documentation>
                                                    An optional parameter used with the
                                                    optional replay feature to indicate the
                                                    newest notifications of interest. If
                                                    stop time is not present, the
                                                    notifications will continue until the
                                                    subscription is terminated. Must be
                                                    used with startTime.
                                                </xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                        <xs:simpleType name="streamNameType">
                            <xs:annotation>
                                <xs:documentation>
                                    The name of an event stream.
                                </xs:documentation>
```

```
            </xs:annotation>
            <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:element name="create-subscription"
            type="createSubscriptionType"
            substitutionGroup="rpcOperation">
            <xs:annotation>
                <xs:documentation>
                    The command to create a notification subscription. It
                    takes as argument the name of the notification stream
                    and filter. Both of those options
                    limit the content of the subscription. In addition,
                    there are two time-related parameters, startTime and
                    stopTime, which can be used to select the time interval
                    of interest to the notification replay feature.
                </xs:documentation>
            </xs:annotation>
        </xs:element>
</xs:schema>
```

C. Notification.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:nns="urn:ietf:params:xml:ns:netconf:notification:1.0"
        xmlns:ens="urn:ietf:params:xml:ns:netconf:notification:event:1.0"
        targetNamespace="urn:ietf:params:xml:ns:netconf:notification:1.0"
        name="netconf-notification_1.0.wsdl">

        <import namespace="urn:ietf:params:xml:ns:netconf:notification:event:1.0"
                location="./notification.xsd" />

        <message name="eventNotification">
                <part name="in" element="nns:notification"/>
        </message>

        <portType name="notificationPortType">
                <operation name="sendNotification">
                        <input message="nns:eventNotification"/>
                </operation>
        </portType>

        <binding name="notificationBinding" type="nns:notificationPortType">
                <SOAP:binding style="document"
                        transport="http://schemas.xmlsoap.org/soap/http"/>

                <operation name="sendNotification">
                        <SOAP:operation/>
                        <input>
                                <SOAP:body use="literal"
                        namespace="urn:ietf:params:xml:ns:netconf:notification:1.0"/>
                        </input>
                </operation>
        </binding>
</definitions>
```

D. Notification.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"
    xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    xml:lang="en">

    <!-- import standard XML definitions -->
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2001/xml.xsd">
        <xs:annotation>
            <xs:documentation>
                This import accesses the xml: attribute groups for the
                xml:lang as declared on the error-message element.
            </xs:documentation>
        </xs:annotation>
    </xs:import>
    <!-- import base netconf definitions -->

    <!--<xs:import namespace="urn:ietf:params:xml:ns:netconf:base:1.0"
        schemaLocation="http://www.iana.org/assignments/xml-registry/schema/netconf.xsd"/> -->
    <!-- ********** One-way Operation ********************-->
    <!-- <Notification> operation -->
    <xs:complexType name="NotificationContentType"/>
    <xs:element name="notificationContent"
        type="NotificationContentType" abstract="true"/>
    <xs:complexType name="NotificationType">
        <xs:sequence>
            <xs:element name="eventTime" type="xs:dateTime">
                <xs:annotation>
                    <xs:documentation>
                        The time the event was generated by the event source.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element ref="notificationContent"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="notification" type="NotificationType">
        <xs:annotation>
            <xs:documentation>
                Sends Asynchronous notification.
            </xs:documentation>
        </xs:annotation>
    </xs:element>
</xs:schema>
```